LEARNING MADE EASY

GitHub®

for dummies®
A Wiley Brand

- Create a repository for your coding projects
- Personalize your workflow with GitHub tools and integrations
- Collaborate on open source software projects

Sarah Guthals, PhD

Phil Haack

Former Engineering Leaders at GitHub

GitHub®

by Sarah Guthals, PhD
and Phil Haack

for dummies®
A Wiley Brand

GitHub® For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2019 by John Wiley & Sons, Inc., Hoboken, New Jersey

Media and software compilation copyright © 2019 by John Wiley & Sons, Inc. All rights reserved.

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. Github is a registered trademark of Github, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE

SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit `https://hub.wiley.com/community/support/dummies`.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at `http://booksupport.wiley.com`. For more information about Wiley products, visit `www.wiley.com`.

Library of Congress Control Number: 2019938575

ISBN 978-1-119-57267-1 (pbk); ISBN 978-1-119-57266-4 (ePDF); ISBN 978-1-119-57265-7 (epub)

GitHub® For Dummies®

To view this book's Cheat Sheet, simply go to www.dummies.com and search for "GitHub For Dummies Cheat Sheet" in the Search box.

Table of Contents

- **Cover**
- **Introduction**
  - About This Book
  - Foolish Assumptions
  - Icons Used in This Book
  - Beyond the Book
  - Where to Go from Here
- **Part 1: Getting Started with GitHub.com**
  - **Chapter 1: Understanding the Git in GitHub**
    - Introducing GitHub
    - Understanding Version Control
    - Git Version Control
    - Git's Place on GitHub
    - Signing Up for GitHub.com
    - Personalizing Your GitHub.com Account

Discovering Helpful Resources

Chapter 2: Setting Up Your Collaborative Coding Environment

Exploring GitHub.com

Understanding Your Profile

Getting to Know GitHub Desktop

Setting up GitHub Desktop

Introducing Atom

Part 2: Starting Your First Solo Project

Chapter 3: Introducing GitHub Repositories

Setting Up a Repository

Exploring Your Repository

Modifying README.md

Merging a Pull Request

Using Issues and Project Boards

Chapter 4: Setting Up a GitHub Website Repo

Introducing GitHub Pages

Turning a Project Repo into a Website

Setting Up a Personal Website Repo

Creating Issues for Your Website

Setting Up Your Local Environment

Finding Resources for GitHub Pages

Chapter 5: Creating a Website with GitHub

Pages

Jumping into an Existing GitHub Project

Preparing Your Contribution

Building Your Personal Website

Part 3: Contributing to Your First Project

Chapter 6: Forking GitHub Repositories

Introducing Forking

Cloning, Forking, and Duplicating

Cloning a Repository

Forking a Repository

Chapter 7: Writing and Committing Code

Creating a Repository

Writing Code

Creating a Commit

Writing a Good Commit Message

Committing Code with GitHub Desktop

Using GitHub Conventions in Commit Messages

Committing Code from Your Editor

Chapter 8: Working with Pull Requests

Understanding a Pull Request

Pushing Code to GitHub

Opening a Pull Request

Writing a Great Pull Request

Reviewing a Pull Request

Reading More About Pull Requests

Part 4: Manage and Contribute to Large Projects

Chapter 9: Exploring and Contributing to OSS

Exploring GitHub

Finding Places to Contribute

Surveying a Project for Contribution

Setting Contributor Expectations

Keeping Tabs on a Project

Chapter 10: Starting Your Own OSS

Creating an Open Source Repository

Making a Repository Public

Enforcing a Code of Conduct

Writing a README.md File

Writing Good Documentation

Managing Issues

Ending Your Project

Chapter 11: Inner-Source Your Code on GitHub

Why Code in Private?

Using GitHub Organizations

Making the Most of Your Teams

Best Practices for Inner-Sourcing

Part 5: Make GitHub Work For You

Chapter 12: Collaborating Outside of GitHub

Chatting It Up
Getting Trello and GitHub Integrated
Managing Notifications with Octobox

Chapter 13: GitHub Workflow Integrations

Using GitHub for Atom
Using GitHub for Visual Studio Code
Using GitHub for Unity
Using GitHub for Visual Studio
Using GitHub for XCode
Using GitHub for IntelliJ

Chapter 14: Personalizing GitHub

Using Browser Extensions
GitHub Apps and Probot
Taking Action with GitHub Actions

Part 6: The GitHub Ecosystem

Chapter 15: Exploring the GitHub Marketplace

Introducing the GitHub Marketplace
Listing Your App on the Marketplace
Considering Common Apps to Install

Chapter 16: GitHub and You

Understanding Your GitHub Profile
Starring Repositories
Following Users
Chapter 17: Attending Events
Exploring Types of Events
Knowing What to Expect at Events
Becoming Familiar with GitHub Events
Speaking at Events
Finding Funding for Events
Part 7: The Parts of Tens
Chapter 18: Ten Ways to Level Up on GitHub
Trial and Error
GitHub Help Docs
GitHub Learning Labs
GitHub In-Person Training
Project-Specific Documentation
External Community Places
Online Coding Tutorials
Online Courses and Tutorials
Blogs and Twitter
Community Forum
Chapter 19: Ten Ways to Improve Your Development Workflow
Drafting Pull Requests

- Git Aliases
- Run Tests Automatically
- Take Breaks
- Prototype User Interfaces
- Scaffold Apps with Yeoman
- Chrome Web Developer Tools
- StackOverflow
- Code Analysis Tools
- Project Boards

Chapter 20: Ten Tips for Being an Effective Community Member

- Be Respectful and Kind
- Report Bad Behavior
- Write Good Bug Reports
- Be Responsive
- Submit Pull Requests to Correct Documentation
- Document Your Own Code
- Give Credit Where It's Due
- Help Get the Word Out
- Be Proactive and Mentor Others
- Contribute Outside of GitHub

Index

About the Authors

Advertisement Page

Connect with Dummies

End User License Agreement

List of Illustrations

Chapter 1

FIGURE 1-1: Example workflow for Git branches.

FIGURE 1-2: The GitHub.com home page when you’re logged in.

FIGURE 1-3: The Personal Settings page.

FIGURE 1-4: Account settings.

FIGURE 1-5: The Email settings.

FIGURE 1-6: The Notifications settings.

FIGURE 1-7: The Billing settings.

FIGURE 1-8: The place to create SSH and GPG keys.

FIGURE 1-9: The Saved replies settings.

Chapter 2

FIGURE 2-1: The home page of GitHub.com.

FIGURE 2-2: The top menu bar of GitHub.com.

FIGURE 2-3: ProTip found on the Pull request page.

FIGURE 2-4: Curated list of repositories on GitHub.com.

FIGURE 2-5: My profile.

FIGURE 2-6: The GitHub Desktop application default view.

FIGURE 2-7: The Sign in dialog box for the GitHub Desktop application.

FIGURE 2-8: The Atom application default view.

FIGURE 2-9: the Squirrel icon takes you to the About page where you can update A...

FIGURE 2-10: Keyboard bindings for toggling the Git and GitHub tabs.

FIGURE 2-11: The Atom Settings page.

Chapter 3

FIGURE 3-1: The page to create a new repository.

FIGURE 3-2: The home page of my HelloWorld repository.

FIGURE 3-3: The Insights tab.

FIGURE 3-4: The Code tab.

FIGURE 3-5: The page to add a file to the `HelloWorld` repository.

FIGURE 3-6: The Open a pull request page.

FIGURE 3-7: The pull request description page for an open pull request.

FIGURE 3-8: The `README.md` file in `edit` mode.

FIGURE 3-9: The `README.md` file in `diff` mode.

FIGURE 3-10: The `README.md` file in `diff` mode with the headshot added.

FIGURE 3-11: My pull request page.

FIGURE 3-12: The diff in split view.

FIGURE 3-13: The dialog box to confirm the merge of the pull request.

FIGURE 3-14: The merge confirmation for a pull request.

FIGURE 3-15: The page to create a new project.

FIGURE 3-16: A blank kanban project board template.

FIGURE 3-17: The automation settings for the To do column on a kanban project.

FIGURE 3-18: The page that shows all issues for the `HelloWorld` repository.

FIGURE 3-19: A new issue linked to the project Tracking Changes to HelloWorld.

FIGURE 3-20: The description page of an open issue.

FIGURE 3-21: The project board with an open issue and an open pull request.

FIGURE 3-22: The diff of `README.md` showing the added text.

Chapter 4

FIGURE 4-1: The GitHub Pages section under repository settings.

FIGURE 4-2: The GitHub Pages website theme chooser.

FIGURE 4-3: The issue list for the website repository.

FIGURE 4-4: The project board for the website repository.

FIGURE 4-5: An overview of GitHub Desktop.

FIGURE 4-6: An overview of Atom.

Chapter 5

FIGURE 5-1: Places to find GitHub repos on the GitHub.com home page.

FIGURE 5-2: Commit changes box when you have write permissions on a repo.

FIGURE 5-3: Warning and commit box when you have read-only permissions on a repo...

FIGURE 5-4: The place to start a review of a pull request.

FIGURE 5-5: The button to create a new branch in GitHub Desktop for a specific r...

FIGURE 5-6: The branch selector in Atom showing that the correct branch is check...

FIGURE 5-7: Editing a file in Atom.

FIGURE 5-8: Staging and committing changes in Atom.

FIGURE 5-9: GitHub.com indicates when a branch is ready to be merged.

Chapter 6

FIGURE 6-1: GitHub.com error message if you don’t have edit permissions on a rep...

FIGURE 6-2: A forked repo on GitHub.com.

FIGURE 6-3: Original, upstream repo detecting a new branch from a forked repo.

FIGURE 6-4: Forked repo detecting a new branch.

FIGURE 6-5: Forked repo detecting a new pull request with the option for upstrea...

FIGURE 6-6: Pull request without the option to merge.

FIGURE 6-7: Pull request with the option to merge.

FIGURE 6-8: Push permission error message in Atom.

FIGURE 6-9: Push permission error message in GitHub Desktop.

Chapter 7

FIGURE 7-1: An alert message from our running code.

FIGURE 7-2: A list of commit messages, some with emojis in the summary.

FIGURE 7-3: GitHub Desktop opened to the `best-example` repository.

FIGURE 7-4: The Publish dialog box used to publish a repository to GitHub.com.

FIGURE 7-5: The Changes View showing the uncommitted changes.

FIGURE 7-6: The diff view with one change selected.

FIGURE 7-7: The Changes tab after a partial commit.

FIGURE 7-8: The emoji picker listing emojis.

FIGURE 7-9: A list of issues with the word “alert” in them.

FIGURE 7-10: A list of potential coauthors.

FIGURE 7-11: The newly created commit.

Chapter 8

FIGURE 8-1: Repository home page with recently pushed branches listed.

FIGURE 8-2: The Open a pull request page.

FIGURE 8-3: List of potential reviewers with two reviewers selected.

FIGURE 8-4: A created pull request.

FIGURE 8-5: Details section collapsed.

FIGURE 8-6: Details section expanded.

FIGURE 8-7: Uploading an image to a pull request.

FIGURE 8-8: Message requesting your review on this pull request.

FIGURE 8-9: Pull request with failed checks.

FIGURE 8-10: Pull Request Comment form for a line of code.

FIGURE 8-11: Pending comment in a pull request review.

FIGURE 8-12: Comment form with the Suggest changes button.

FIGURE 8-13: Comment with a suggested change.

FIGURE 8-14: Comment with a suggested change.

FIGURE 8-15: Comment with a suggested change.

Chapter 9

FIGURE 9-1: The Explore page on GitHub.

FIGURE 9-2: The Discover page on GitHub with repository recommendations.

FIGURE 9-3: List of topics for a repository.

FIGURE 9-4: A list of suggested topics.

FIGURE 9-5: Topic page for NodeJS.

FIGURE 9-6: Profile page for Phil, one of the authors.

FIGURE 9-7: Up for grabs filter.

FIGURE 9-8: Visual Studio Code Issues filtered by the `good first issue` label.

Chapter 10

FIGURE 10-1: Home page for our soon-to-be open source repository.

FIGURE 10-2: Choose a license template button magically appears.

FIGURE 10-3: The license chooser page.

FIGURE 10-4: The license information page.

FIGURE 10-5: Danger zone: where you can make a private repo public.

FIGURE 10-6: Confirmation to convert a private repo public.

FIGURE 10-7: The set of interaction limits.

FIGURE 10-8: Profile page with link to block or report user.

FIGURE 10-9: Dialog box with Block user and Report abuse buttons.

FIGURE 10-10: Serving a GitHub Pages site from the docs folder.

FIGURE 10-11: Unlabeled issues filter.

FIGURE 10-12: Choosing an issue template.

FIGURE 10-13: Choosing an issue template.

FIGURE 10-14: Choose an issue template to create an issue.

FIGURE 10-15: A new issue with the Bug report template filled in.

FIGURE 10-16: Creating a new saved reply.

FIGURE 10-17: Selecting a saved reply in a comment box.

FIGURE 10-18: The button to archive a repository.

FIGURE 10-19: The button to archive a repository.

FIGURE 10-20: The transfer repository dialog box.

Chapter 11

FIGURE 11-1: Inviting someone to join your organization.

FIGURE 11-2: Invitation request from the invitee’s account.

FIGURE 11-3: Repositories associated with the Microsoft organization.

FIGURE 11-4: A list of members of an organization.

FIGURE 11-5: An overview of a member of an organization.

FIGURE 11-6: Creating a project board linked with multiple repositories.

FIGURE 11-7: Settings for organization project boards.

FIGURE 11-8: Options for blocking an individual.

FIGURE 11-9: Options for blocking an individual.

FIGURE 11-10: Repository access for a specific team.

FIGURE 11-11: Requiring at least one approving review from a CODEOWNER.

FIGURE 11-12: CODEOWNERS automatically added to a pull request.

FIGURE 11-13: GitHub must have read access to your repository to give security a...

FIGURE 11-14: Allow GitHub read access to your repository for various data servi...

FIGURE 11-15: The network graph for the GitHub package for Atom open source proj...

Chapter 12

FIGURE 12-1: Confirmation page for the GitHub app.

FIGURE 12-2: Choose which channels the GitHub app is enabled in.

FIGURE 12-3: Slack message with a prompt to connect your GitHub account.

FIGURE 12-4: Authorize the Slack app for GitHub.

FIGURE 12-5: Configuring the Slack app on GitHub.

FIGURE 12-6: Summary of available GitHub commands on Slack.

FIGURE 12-7: Slack dialog to create an issue on GitHub.

FIGURE 12-8: Slack message with information about a newly created GitHub issue.

FIGURE 12-9: Power-Ups section of the menu.

FIGURE 12-10: Authorize Trello on GitHub.

FIGURE 12-11: Trello card with a GitHub power-up.

FIGURE 12-12: Selecting the issue to attach.

FIGURE 12-13: Trello card with an issue and pull request attached.

FIGURE 12-14: Front of a card with an issue and pull request attached.

FIGURE 12-15: GitHub issue with a link to the Trello board.

FIGURE 12-16: GitHub app download button and Pricing options for Octobox.

FIGURE 12-17: Octobox inbox for the author.

Chapter 13

FIGURE 13-1: The GitHub tab when no project is open in Atom.

FIGURE 13-2: The GitHub tab when a GitHub project is open in Atom, but you have ...

FIGURE 13-3: The GitHub tab showing all open pull requests on that repository.

FIGURE 13-4: The detailed view of a pull request with the Checkout button.

FIGURE 13-5: The GitHub tab showing the current pull request at the top.

FIGURE 13-6: The GitHub tab showing the current pull request at the top.

FIGURE 13-7: GitHub issue details opened in Atom.

FIGURE 13-8: GitHub extension for VS Code in the extension marketplace.

FIGURE 13-9: Three ways to start the GitHub sign-in process.

FIGURE 13-10: Interacting with pull requests in VS Code.

FIGURE 13-11: Adding an inline comment in VS Code.

FIGURE 13-12: GitHub for Unity in the Asset Store.

FIGURE 13-13: Installing the GitHub for Unity extension.

FIGURE 13-14: Signing in to GitHub inside of Unity.

FIGURE 13-15: Publishing your project to GitHub.com.

FIGURE 13-16: The GitHub for Visual Studio extension in the Visual Studio Market...

FIGURE 13-17: The Team Explorer pane with the GitHub Connect section.

FIGURE 13-18: The Team Explorer pane with additional GitHub project options.

FIGURE 13-19: A list of pull requests on the GitHub pane.

FIGURE 13-20: Interacting with a pull request in Visual Studio.

FIGURE 13-21: Signing in to GitHub inside of XCode.

FIGURE 13-22: List of GitHub repositories you have access to clone from inside X...

FIGURE 13-23: Clone from a git project to get started integrating GitHub into In...

FIGURE 13-24: List of GitHub repositories you have access to clone from inside I...

FIGURE 13-25: Viewing a pull request from inside of IntelliJ.

FIGURE 13-26: Creating a pull request from inside of IntelliJ.

Chapter 14

FIGURE 14-1: Configuring the Refined GitHub extension.

FIGURE 14-2: Taking a short video with GitHub Selfie.

FIGURE 14-3: The README page of a Glitch Probot app.

FIGURE 14-4: Updating the event subscriptions for the GitHub app.

FIGURE 14-5: The why not both app in action.

FIGURE 14-6: The landing page for GitHub Actions workflows.

FIGURE 14-7: Committing a simple workflow.

Chapter 15

FIGURE 15-1: Your Settings page on GitHub.com where you can add payment informat...

FIGURE 15-2: Your applications that you can turn into Marketplace listings.

FIGURE 15-3: Filling out a form to list an app.

FIGURE 15-4: Steps to fill out a Marketplace submission.

FIGURE 15-5: AppVeyor CI app example on the GitHub for Visual Studio repository.

FIGURE 15-6: Codecov app example on the GitHub package for Atom repository.

FIGURE 15-7: Greenkeeper app example on the GitHub package for Atom repository.

Chapter 16

FIGURE 16-1: The profile page for Haacked.

FIGURE 16-2: Changing your status message.

FIGURE 16-3: Staff badge on a GitHub profile.

FIGURE 16-4: Displaying private contributions and the activity graph.

FIGURE 16-5: Profile visualization for one of the authors.

Chapter 18

FIGURE 18-1: Giving the GitHub Learning Lab app access to a specific repository.

FIGURE 18-2: Effective pull request review feedback on the VS Code open source p...

Chapter 19

FIGURE 19-1: Drafting a pull request.

FIGURE 19-2: The ready for review button.

Introduction

Welcome to the world of collaborative coding! Whether you're just starting your coding journey, building fairly complex programs, or building with a team of people, this book guides you in using one of the most used tools for collaborative code-writing; GitHub.com. With more than 31 million users and over 100 million repositories (projects) hosted, GitHub.com is the No. 1 place to build and collaborate on code.

About This Book

Though you spend many hours sitting at your computer, alone, debugging and writing code, the ideal coding team includes more than just you. Hundreds of developers spent more than four years building World of Warcraft before its first release in 2004. Although occasionally you can build a big hit like Flappy Bird alone, in a couple of days, the norm for software development is that you will work with other coders, designers, testers, user experience experts, product managers, and sometimes hardware engineers to bring something to the hands of users.

When you're first starting out on complex coding projects, understanding effective ways to collaborate can be daunting. This book introduces you to the world of open source development (the epitome of collaboration), as well as effective ways to work with one other person — or even yourself over the course of many years! (I don't know about you, but Sarah from three years ago knows stuff that Sarah from today can't remember, and Sarah from today has more experience than Sarah from three years ago.)

GitHub For Dummies is written as a reference guide. Each part introduces you to a different aspect of collaborative coding. So if you're experienced in using GitHub, but you're new to the open source community, you can jump to Part 5 and skip some of the GitHub basics.

As you explore each part of this book, keep the following points in mind:

- Words that are being defined appear in *italic*.

- Code and URLs (web addresses) are shown in `monofont`.
- Command sequences using onscreen menus use the command arrow. For example, when working in Scratch, you can open a new project as follows: From the menu bar, choose File ⇒ New.
- The figures you see in this book use Mac and Chrome. We provide some tips when what you see on a Windows PC may be different, but you should see the same things, regardless of which Internet browser you use.

Foolish Assumptions

In this book, I make some assumptions that very well may be foolish, about you, your coding experience, and your goals.

- You're interested in and have had some experience with coding. You don’t have to be an expert coder, but you have made a Hello World application (or the equivalent) in at least one programming language.
- You have patience and determination and are resourceful. When you’re presented with a challenge, you can Google your way to finding a solution. This book guides you through GitHub.com as it exists at the time of writing it, but new features and workflows are being created, and part of your collaborative coding journey is to discover how to use those new features as they become available.
- You have experience with a keyboard and mouse on either Mac or Windows PC and have access to one of those machines.
- You’re capable of using an Internet browser, such as Safari, Chrome, or Firefox, and you can type a URL to access a website, such as GitHub.com.
- You know how to install applications on your computer. Although we guide you through anything that is unique to the setup, you should know how to download and install an application without step-by-step guidance.

Icons Used in This Book

Throughout the margin of this book are small images, known as *icons*. These icons mark important tidbits of information:

TIP The Tip icon identifies places where we offer additional tips for making this journey more interesting or clear. Tips can start you on a rabbit hole down another workflow, not covered in this book, or cover some neat shortcuts that you may not have known about.

REMEMBER The Remember icon bookmarks to important ideas to help you work more effectively throughout this book.

WARNING The Warning icon helps protect you from common errors and may even give you tips to undo your mistakes.

Beyond the Book

In addition to what you're reading right now, this product also comes with a free access-anywhere Cheat Sheet that covers common commands and GitHub actions. To get this Cheat Sheet, simply go to `www.dummies.com` and search for "GitHub For Dummies Cheat Sheet" in the Search box.

Other online resources also pair with this book:

- Online articles covering additional topics are available at `www.dummies.com/extras/github`.
- Updates to this book, if any, can be found at `www.dummies.com/extras/github`.
- GitHub Learning Labs are free, guided tutorials that can be installed and found at `https://lab.github.com`.

Where to Go from Here

GitHub is a tool used by millions of developers. The workflows that you discover in this book is just the beginning. As you become a more experienced coder, begin to collaborate on more elaborate projects, or join different companies and teams, you may encounter new workflows that use these tools in different ways. You should feel empowered to explore! Visit `https://help.github.com` or `https://guides.github.com` for guidance and don't forget to follow the blog at `https://blog.github.com/` to stay up to date with all of the new features!

Part 1

Getting Started with GitHub.com

IN THIS PART …

Discover how to use Git on your local computer to track changes in your project.

Sign up for a free GitHub.com account.

Explore GitHub.com resources and features.

Install GitHub Desktop to manage the link between your local and remote projects.

Install the GitHub Atom editor as a lightweight option for coding.

Prepare for creating your own projects and contributing to others.

Chapter 1

Understanding the Git in GitHub

IN THIS CHAPTER

- **Getting familiar with GitHub**
- **Discovering Git**
- **Signing up with GitHub.com**
- **Exploring helpful resources**

Whether you're an experienced coder or a newbie starting out, learning how to work with others on code is critical to succeeding in the software industry. Millions of people around the world work together to build software, and GitHub is one of the largest tools to support a collaborative workflow. This chapter introduces you to the core tools you need to write code with other people.

Introducing GitHub

GitHub creates an environment that allows you to store your code on a remote server, gives you the ability to share your code with other people, and makes it easy for more than one person to add, modify, or delete code to the same file and project, while keeping one source of truth for that file (phew!). So what does that all actually mean? One of my favorite ways of explaining GitHub.com to folks who are new to the tool is to compare it to Google Docs — a place online where you can write code with other people and not have to email different versions back and forth.

What makes GitHub work behind the scenes is Git.

Understanding Version Control

Version control systems (also known as source control management, or SCM) are software that keep track of each version of each file in a coding project, a timestamp for when that version was created, and the author of those changes.

TIP Writing code is an iterative process. For example, when you're building a website, you first may want to get some basic structure up before adding all your content. The best thing to do is to create a version of your website each time you have something that works. That way, as you experiment with the next piece, if something breaks, you can just go back to your previous version and start over.

SCMs enable coders to make mistakes without worrying that they'll have to completely start over. Think of it like being able to click Undo, but instead of undoing each key press, you can undo an entire piece of the project if you decide you don't like it or it doesn't work.

The basic workflow of coding with version control system support is as follows:

1. **Create a project, typically in a folder on your computer.**
2. **Tell your version control system of choice to track the changes of your project/folder.**
3. **Each time your project is in a working state, or you're going to walk away from it, tell your version control system of choice to save it as the next version.**
4. **If you ever need to go back to a previous version, you can ask your version control system to revert to whichever previous version you need.**

You can use a version control system if you're working alone on your own computer, but it gets even more interesting when you begin working with other people. (For more on working with other people, see the section "Git version control," later in this chapter).

For more information about version control, visit `https://git-`

`scm.com/book/en/v2/Getting-Started-About-Version-Control`.

Git Version Control

GitHub, as the same would suggest, is built on Git. Git is a type of version control system, and it is free and open source, which means that anyone can use it, build on top of it, and even add to it.

GitHub products make using Git easy, but if you're curious, you can also use Git to track your solo projects on your computer. You can find a brief introduction to local Git commands for solo projects in the section "Try simple Git on the terminal".

Try simple Git on the terminal

With the help of Git for Windows, using the terminal on Mac, Windows, or Linux computers is exactly the same. A *terminal* is an application that enables you to interact with your computer in a text-based way — in other words, instead of double-clicking and dragging, you type commands to navigate your computer.

If you're on Mac or Linux, a terminal is already installed on your computer. If you're using a Windows computer, install Git for Windows, which is what this section assumes you've installed. Just go to `https://gitforwindows.org` and click Download to gain access to Git Bash, an emulator that allows you to interact with Git just like you would on a Linux or Mac terminal. You also get Git GUI, which gives you a user interface for almost all Git commands you might type into Git Bash, and shell integration so that you can quickly open Git Bash or Git GUI from any folder.

TIP Many developers on Windows prefer to use PowerShell as their terminal environment. You can use Git within PowerShell, but setting that up properly is outside the scope of this book. However, Phil, one of the authors, has a handy guide to setting this up at `https://haacked.com/archive/2011/12/13/better-git-with-powershell.aspx`.

First, find the Terminal application:

- On Mac, you can click the magnifying glass at the top right of your desktop, type **Terminal**, select the terminal from the list of applications, and press Enter or click it.
- On Linux, press Ctrl-Alt-T all at the same time, and the terminal window opens.
- On Windows, click the Windows menu in the bottom right of your desktop, search **Git Bash,** select the Git Bash application from the list of search results, and press Enter or click it.

When the application opens, type `git --version` in the terminal. If you have Git installed, you should see a version number, as shown in the following code (the `$` should already be on the line, so you do not need to type it). Otherwise, you can follow the instructions on `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`.

WARNING When using the command line, you have to be very careful about exactly what you're typing. In the following code, the first instruction is for you to type `git --version`. You should note that a space appears between `git` and the rest of the instruction but no other spaces. You should also note the two dashes before the word `version`. They can be easy to miss, so be careful!

For Mac or Linux, you should see something like this:

```
$ git --version
git version 2.16.3
$
```

For Windows, you should see something like this:

```
$ git --version
git version 2.20.1.windows.1
$
```

Next, using the terminal, go to your desktop and create a new folder called git practice. To do this, you should type the following commands:

```
$ cd ~/Desktop
$ mkdir git-practice
$ cd git-practice
$
```

If you type `pwd`, you should see that you are now in the folder git-practice, which is on your desktop. It might look something like this:

```
$ pwd
$ /Users/sguthals/Desktop/git-practice
$

Now, you can tell git to track this folder using the init
   command.
$ git init
Initialized empty Git repository in /Users/sguthals/Desktop/
   git-practice
$
```

Then make sure that you have a clean folder. You can check with the `status` command:

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to
   track)
$
```

Then, you can create a file to have Git start tracking and confirm the file is in the folder:

```
$ echo "practicing git" > file.txt
$ ls
file.txt
$
```

On Mac, you can open this folder in a Finder with the `open <path>` command:

```
$ open .
$
```

On Linux, you can open this folder with the `nautilus <path>` command:

```
$ nautilus .
$
```

On Windows, you can open this folder with the `explorer <path>` command:

```
$ explorer .
$
```

In this example, we put `.` as the `<path>` for each command. `.` tells the terminal to open the current folder. You could also use a different path with these commands to open other folders.

After the folder is open, double-click the file called `file.txt`, and the file opens with TextEdit on Mac, gedit on Linux, and Notepad on Windows. You can see that the words “practicing git” are actually there.

Close the file. Now, you can tell Git that you want to save this as a particular version. Back in the terminal:

```
$ git add file.txt
$ git commit -m "Adding my file to this version"
[master (root-commit) 8d28a21] Adding my file to this version
 1 file changed, 1 insertion(+)
 Create mode 100644 file.txt
$ git status
On branch master
nothing to commit, working tree clean
$
```

You can make a change to your file in the text file. Open the file again, change the text to say “Hi! I'm practicing git today!” and then click File ⇒ Save and close the text application.

When you go back to the Terminal to check the status of your project again, you should see that Git has noticed that the file has changed:

```
$ git status
On branch master
Changed not staged for commit:
  (use "git add <file…" to update what will be committed)
  {use "git checkout -- <file>…" to discard changed in working
   directory)

        modified:    file.txt
```

```
no changed added to commit (use "git add" and/or "git commit -
    a")
$
```

`Commit` this version of your file again and notice that Git recognizes that everything has been saved to a new version:

```
$ git add file.txt
$ git commit -m "I changed the text"
[master 6d80a2a] I changed the text
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git status
On branch master
nothing to commit, working tree clean
$
```

TIP If your terminal starts to get too cluttered, you can type `clear` to clear some space and make it more visually appealing. Don't worry; you can always scroll up and see everything you typed earlier!

Say that you actually want to go see the original change; when you added “practicing git”. First, get the `log` of all the `commits` you have made:

```
$ git log
commit 6d80a2ab7382c4d308de74c25669f16d1407372d (HEAD ->
    master)
Author: sguthals <sguthals@github.com>
Date:   Sun Dec 9 08:54:11 2018 -0800

    I changed the text

commit 8d28a21f71ec5657a2f5421e03faad307d9eec6f
Author: sguthals <sguthals@github.com>
Date:   Sun Dec 9 08:48:01 2018 -0800

    Adding my file to this version
$
```

Then ask Git to show you the first `commit` you made (the bottom most one). Make sure that you're typing your unique `commit hash`. In this book, the hash starts with `8d28a2`. Make sure you type the entire hash that appears in your Git log:

TIP Instead of typing the entire hash (and possibly having a typo), you can highlight the hash with your mouse, right-click and choose copy, and then after `git checkout`, you can right-click and choose Paste. Using the keyboard shortcuts Ctrl + C or ⌘-C doesn't work

```
$ git show 8d28a21f71ec5657a2f5421e03faad307d9eec6f
commit 8d28a21f71ec6567a2f5421e03faad307d9eec6f
Author: sguthals <sarah@guthals.com>
Date:   Sun Dec 9 08:48:01 2018 -0800

    Adding my file to this version

diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..849a4c7
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+practicing git
$
```

You can see that `practicing git` was added to the file in that original commit.

For more information on how to use git on the command line, check out the following resources:

- The GitHub Git Cheat Sheet at `https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf`
- The Visual Git Cheat Sheet at `http://ndpsoftware.com/git-cheatsheet.html`
- The Git Docs page at `https://git-scm.com/doc`

Another couple resources for learning and understanding Git is `https://learngitbranching.js.org` and `http://git-school.github.io/visualizing-git`, which enable users on Windows to experience a similar workflow because they're visualizations hosted on a website. The first link, learninggitbranching.js.org, is a good self-guided set of exercises, while the second link, git-school, is best used for folks who have a decent understanding of Git and want to explore what will happen in different scenarios, or for folks who have a more expert Git user guiding them.

Git branching by collaborator

Git is different from other version control systems because it has fast branching, shown in Figure 1-1. *Branching* is a Git function that essentially copies code (each `branch` is a copy of the code), allows you to make changes on a specific copy, and then merges your changes back into the main (`master`) branch.

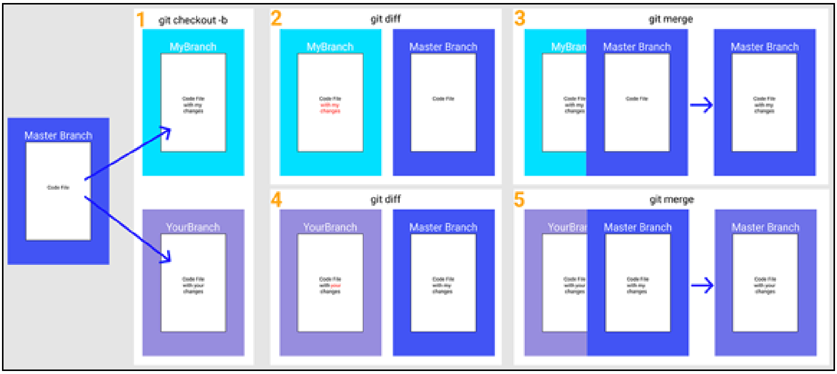

Master Branch
Code File
1
git checkout -b
MyBranch
Code File with my changes
YourBranch
Code File with your changes
2
git diff
MyBranch
Code File with my changes
Master Branch
Code File
3
git merge
MyBran
Master Branch
Code File
Master Branch
Code File with my changes
4
git diff
YourBranch
Code File with your changes
Master Branch
Code File with my changes
5
git merge
YourBra
Master Branch
Code File with my changes
Master Branch
Code File with your changes

FIGURE 1-1: Example workflow for Git branches.

When you're writing code, you will add files and commit changes to your `master` branch. Figure 1-1 outlines a specific workflow where two people are collaborating on the same file. Person 1 creates a new branch called `MyBranch` and makes some changes to the file. Person 2 also creates a new branch called `YourBranch` and makes some changes to the same file. You can see this change in box #1.

You can see the difference (called `diff`) between the `master` branch and `MyBranch` in Figure 1-1 in box #2.

Then, Person 1 `merges` their changes with the `master branch`, as you can see in box #3.

Person 2 has made their own changes, but before merging, they will make sure they have the most updated version of the `master branch`, which now has the changes from Person 1. The `diff` can be seen in box #4. Notice what text is in both files.

Finally, Person 2 acknowledges that their changes will overwrite Person 1's changes and merges their changes with master, making the final version have the changes from Person 2. Box #5 shows this final merge, with the `master branch` having the final changes.

REMEMBER Figure 1-1 shows just one workflow that can exist when more than one person is working on code and is meant to describe branching. You can get a more in-depth overview on Git and branching at `https://git-scm.com`.

Git branching by feature

Another common way to use `branching` is to have each feature that you develop be on a different `branch`, regardless of the collaborator who is building the feature.

You can extend the idea of branching by feature to also have one branch that is your `production` branch. This branch is what your users will see. Then you can have a `development` branch, which is one that you can merge features into without changing what your users see.

This type of branching allows you to build a lot of different features, merge them each into the `development branch`, make sure they all work the way you want, and then `merge` the `development` branch into the `production` branch when you know it's ready for your users.

Git branching for experimentation

You can also create branches to test to see whether something works and then completely throw the `branch` away.

TIP This type of branching can be useful if you want to try a completely new layout of a website, for example. You can create three different branches, each with a different layout.

After you decide which layout you like best, you can simply delete the other two branches and merge the branch with your favorite layout into `master`.

Git's Place on GitHub

GitHub is a host for Git repositories. At some point, it is helpful to place your Git repository in a shared location as both a backup and a place where others can collaborate with you on your code. As a Git host, Git provides all the features of Git in addition to a few extra useful services.

REMEMBER On GitHub.com, projects, or *repositories,* are stored on remote GitHub servers. If you save all your code on GitHub.com and your computer crashes, you can still access it.

Here is a list of some core Git features that GitHub supports:

- **Repository:** Each repository contains all the files and folders related to your project and gives you control of permissions and collaborators' interaction with your code.
- **Clone:** When you want to make changes to your code, you will often want to create a copy, or `clone`, of the project on your local computer. The cloned project is still tightly connected with the version on GitHub.com; it's just your local copy.
- **Fork:** *Forking* a project is when you create your own copy of the entire project. When you `fork` a project, GitHub.com creates a new repository with your copy of all the files. You can still suggest changes back to the original copy, but you can also take your version and go in a new direction.
- **Branches:** GitHub.com supports branching and even provides a useful tool — *pull requests* — to compare the diff between branches and `merge` branches.
- **Commits:** GitHub.com tracks all the `commits` that you push to its servers and gives you an easy interface for browsing the code at different branches and different commits.

Signing Up for GitHub.com

GitHub.com offers unlimited free public and private repositories for individuals. Free private accounts are limited to three collaborators. You can sign up for a paid account to have unlimited collaborators and some Pro features.

Public means that anyone can see your code, clone your code, and therefore use your code. GitHub is a huge supporter of *open source software* (OSS) and therefore encourages public, shared code. Open source software is more than just public, shared code (see Part 5). Because every line of code can be traced to an author, you still get credit for what you've written, but the goal is to keep the code available for anyone to use, extend, and explore.

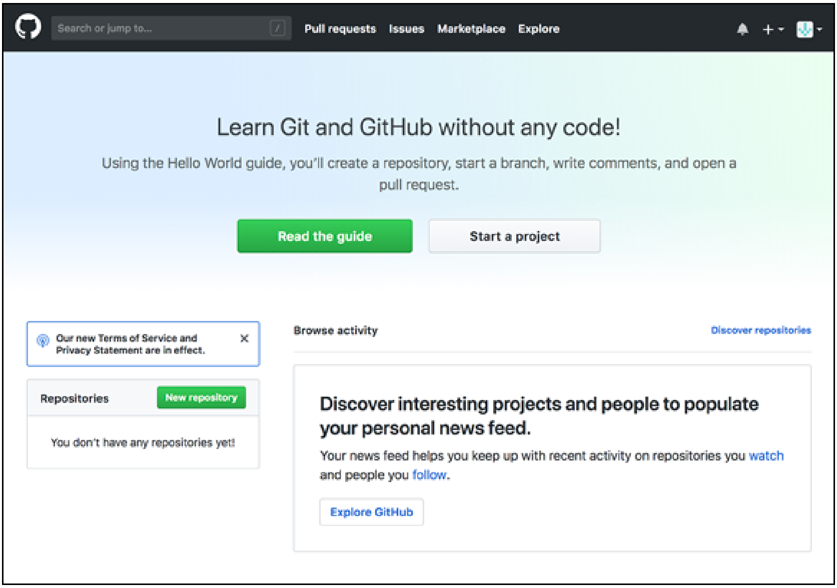
The following steps guide you through signing up for a free, individual GitHub.com account:

1. **Go to GitHub.com and fill out the Sign Up form.**
2. **Choose the plan you want.**

   For the purpose of this book, you can use the Free plan. You can always upgrade to a paid plan later if you decide you want to have more than three collaborators for your private repository and other pro GitHub features.
3. **Complete the brief survey.**

   This survey helps GitHub understand who is using the software and helps them support workflows specific to their users.

   You’re now at the home page, shown in Figure 1-2.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Read the guide
Start a project
Repositories
New repository
Browse activity
Discover repositories
Explore GitHub

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Our new Terms of Service and Privacy Statement are in effect.

You don't have any repositories yet!

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch and people you follow.

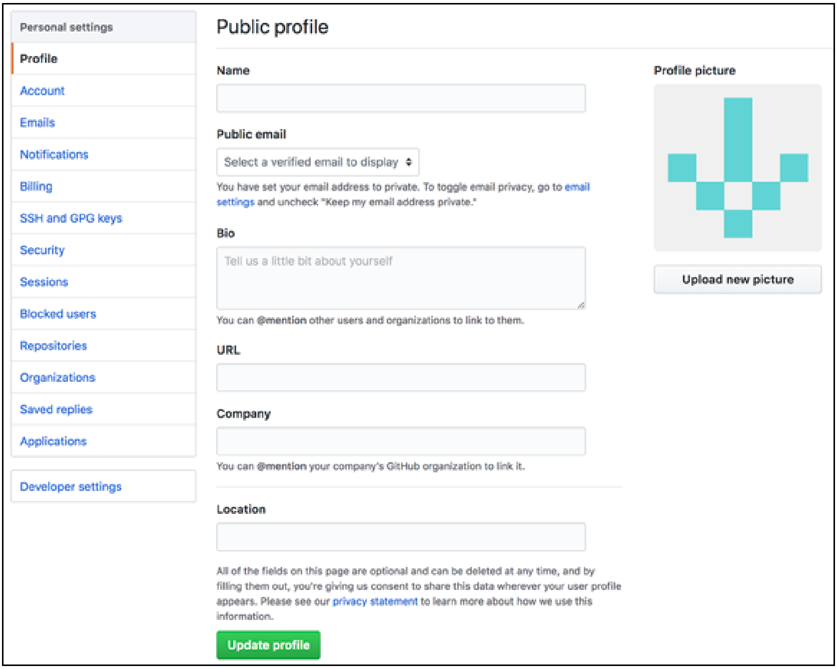
FIGURE 1-2: The GitHub.com home page when you're logged in.

Personalizing Your GitHub.com Account

As you become a more experienced coder, you may want to reference your GitHub.com profile on your resume and job applications. More and more companies care more about your portfolio than a list of degrees or awards. For example, GitHub doesn't require you to provide information on your education as part of the hiring process and instead asks for a link to your GitHub.com profile and/or portfolio.

To complete your GitHub.com profile:

1. **Click the avatar icon in the top right corner and choose Your Profile.**
2. **Click Edit Profile on the page that appears.**
3. **Fill out the form on the Personal Settings page, shown in Figure 1-3.**
4. **Click Update profile when you're finished.**


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Public profile
Name
Public email
Select a verified email to display
Bio
Tell us a little bit about yourself
URL
Company
Location
Profile picture
Upload new picture
Update profile

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

You can **@mention** other users and organizations to link to them.

You can **@mention** your company's GitHub organization to link it.

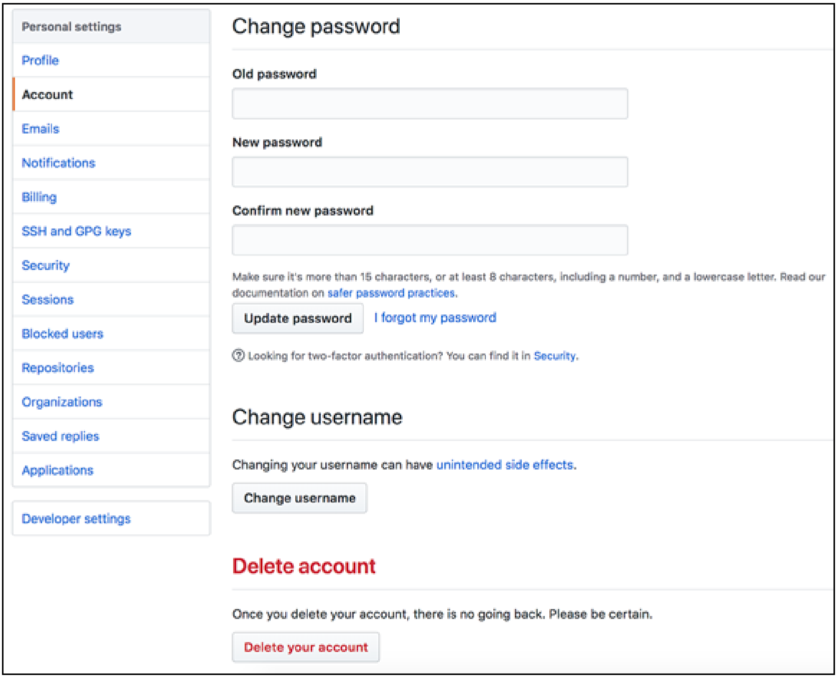
All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.

FIGURE 1-3: The Personal Settings page.

On the Personal Settings page, you can also adjust a number of different settings to continue personalizing your account.

Account

In Account settings, you can change your password, change your username, or delete your account (see Figure 1-4).


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings

Change password

Old password

New password

Confirm new password

Make sure it's more than 15 characters, or at least 8 characters, including a number, and a lowercase letter. Read our documentation on safer password practices.

Update password
I forgot my password

Looking for two-factor authentication? You can find it in Security.

Change username

Changing your username can have unintended side effects.

Change username

Delete account

Once you delete your account, there is no going back. Please be certain.

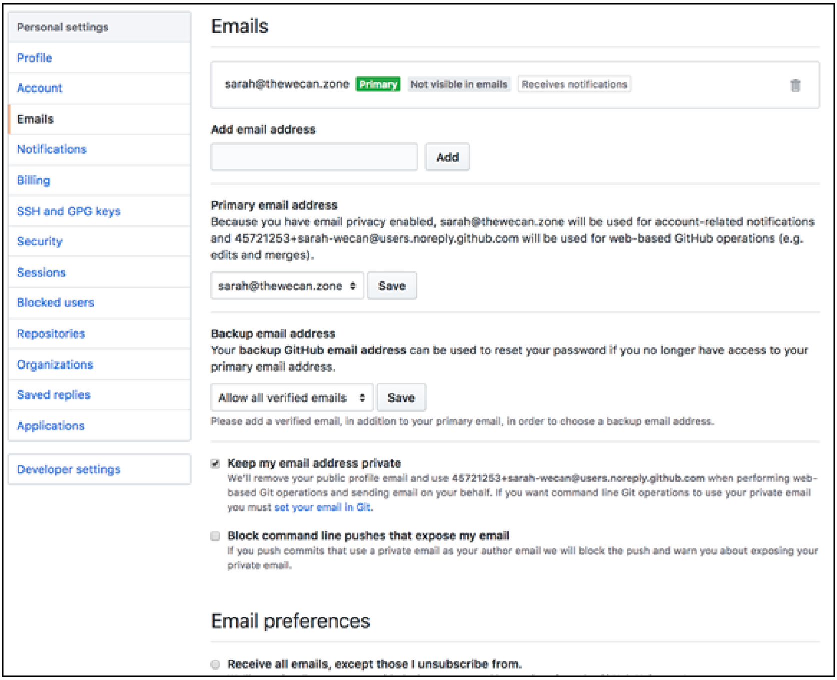
Delete your account

FIGURE 1-4: Account settings.

WARNING Changing your username may cause unintended side effects, so it typically isn't recommended. Just make sure that after you change your username that anything that you need to continue working still does. Follow links, test code, and run your applications again.

Emails

GitHub allows you to link multiple email addresses to your account. Notice that you can add email addresses, keep your email address private, and even block Git commands that may expose your email address (see Figure 1-5).


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Add
Save
Save

Emails

sarah@thewecan.zone Primary Not visible in emails Receives notifications

Add email address

Primary email address
Because you have email privacy enabled, sarah@thewecan.zone will be used for account-related notifications and 45721253+sarah-wecan@users.noreply.github.com will be used for web-based GitHub operations (e.g. edits and merges).

sarah@thewecan.zone

Backup email address
Your **backup GitHub email address** can be used to reset your password if you no longer have access to your primary email address.

Allow all verified emails

Please add a verified email, in addition to your primary email, in order to choose a backup email address.

☑ **Keep my email address private**
We'll remove your public profile email and use **45721253+sarah-wecan@users.noreply.github.com** when performing web-based Git operations and sending email on your behalf. If you want command line Git operations to use your private email you must set your email in Git.

☐ **Block command line pushes that expose my email**
If you push commits that use a private email as your author email we will block the push and warn you about exposing your private email.

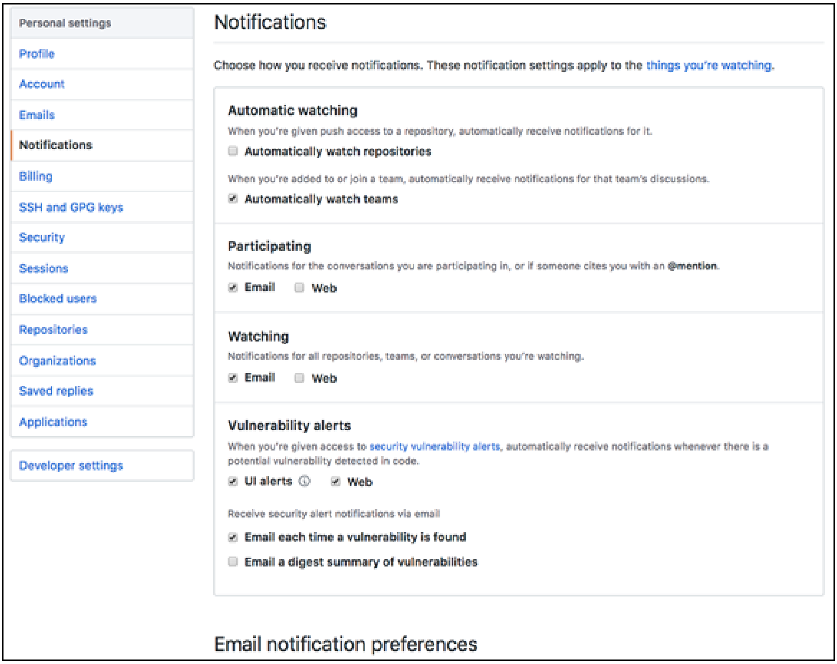
Email preferences

○ **Receive all emails, except those I unsubscribe from.**

FIGURE 1-5: The Email settings.

Notifications

Notifications can get really overwhelming. Though you can choose your level of granularity for receiving notifications per repository, this page creates your default preferences for notifications (see Figure 1-6).



Personal settings
- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Sessions
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

Developer settings

Notifications

Choose how you receive notifications. These notification settings apply to the things you're watching.

Automatic watching

When you're given push access to a repository, automatically receive notifications for it.

☐ Automatically watch repositories

When you're added to or join a team, automatically receive notifications for that team's discussions.

☑ Automatically watch teams

Participating

Notifications for the conversations you are participating in, or if someone cites you with an **@mention**.

☑ Email ☐ Web

Watching

Notifications for all repositories, teams, or conversations you're watching.

☑ Email ☐ Web

Vulnerability alerts

When you're given access to security vulnerability alerts, automatically receive notifications whenever there is a potential vulnerability detected in code.

☑ UI alerts ⓘ ☑ Web

Receive security alert notifications via email

☑ Email each time a vulnerability is found

☐ Email a digest summary of vulnerabilities

Email notification preferences

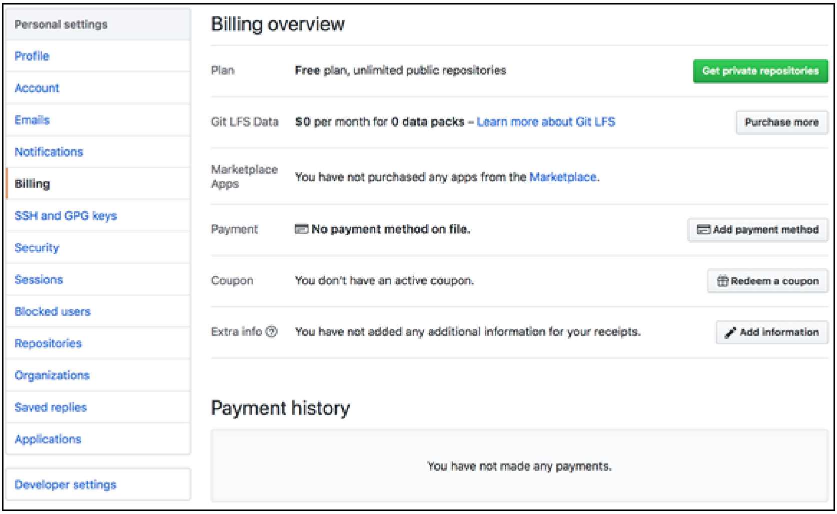
FIGURE 1-6: The Notifications settings.

WARNING We recommend not automatically watching repositories because any kind of activity that happens on any repository that you interact with will start to show up in your inbox, which turns out to not be helpful as you begin collaborating more.

TIP Click the Things you're watching link at the top of the notifications settings page to check to see what you're watching and therefore what notifications you may get from them.

Billing

You can upgrade your plan at any time to include Pro features, such as unlimited collaborators and advanced code review tools. You can make this upgrade happen on the Billing settings page, shown in Figure 1-7. In addition to upgrading your plan, you can also purchase Git LFS data and other Marketplace Apps. You can even put in coupons from your company, school, or affiliated organization.


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Get private repositories
Purchase more
Add payment method
Redeem a coupon
Add information

Billing overview

Plan Free plan, unlimited public repositories

Git LFS Data $0 per month for **0 data packs** – Learn more about Git LFS

Marketplace Apps You have not purchased any apps from the Marketplace.

Payment **No payment method on file.**

Coupon You don't have an active coupon.

Extra info You have not added any additional information for your receipts.

Payment history

You have not made any payments.

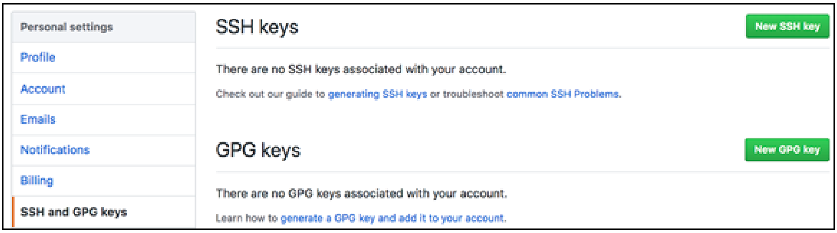
FIGURE 1-7: The Billing settings.

TIP

Git LFS stands for Git Large File Storage. Some software development requires large files, such as game scenes in video game development, to be stored. Without Git LFS, you can upload files as large as 100MB. Anything larger requires Git LFS, which supports files up to a couple GB.

SSH and GPG keys

At some point, you may want to create an SSH or GPG key to encrypt your communication with GitHub and ensure a secure environment. You can do this in your settings, shown in Figure 1-8.



SSH keys

There are no SSH keys associated with your account.

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

GPG keys

There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account.

FIGURE 1-8: The place to create SSH and GPG keys.

TIP SSH keys enable you to connect to GitHub from your local machine without having to put in your username and password each time. GPG keys mark tags and commits that you make as verified, meaning that other people know that it was actually you who push the changes.

TIP Another way to tell Git to remember your credentials is to use a credential helper. GitHub tends to recommend this over using SSH, especially for Windows users. For more information on how to set up this feature, visit `https://help.github.com/articles/caching-your-github-password-in-git`.

Security

Not only should your password be complex, but you should also consider enabling two-factor authentication. *Two-factor authentication* means that when you type the correct password, you're asked to further verify that it is you who is attempting to log in through an app or SMS.

Sessions

Sessions allows you to see every computer address, city, and country where you're logged in or connecting to GitHub.com.

Blocked users

In the Blocked users settings, you can block users from all your repositories.

Repositories

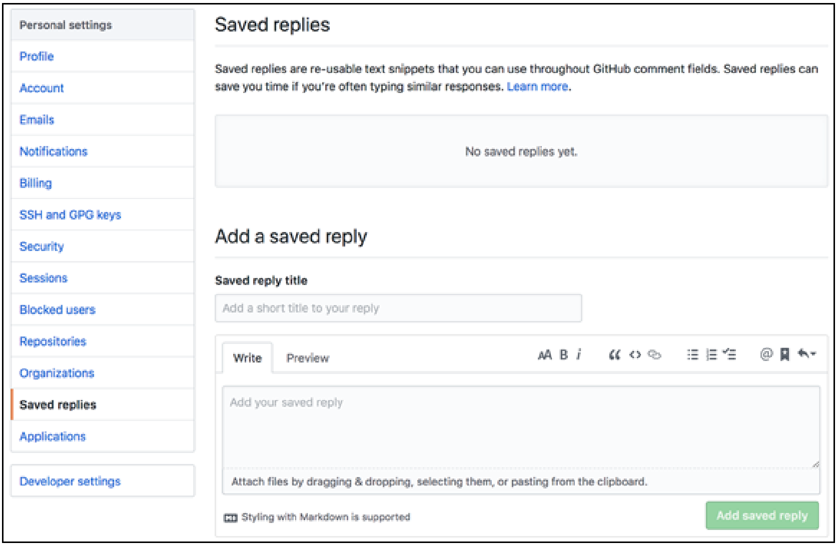
The Repositories section lists all the repositories that you have created or been invited to as a collaborator. You also can leave repositories from this page.

Organizations

Organizations enable you to put GitHub users and repositories under similar settings. For example, you can grant admin rights to all repositories in an organization to the entire organization, instead of having to individually add each person to each repository. Although Organizations is out of the scope of this book, you can read about them on the GitHub Help page at `https://help.github.com/articles/about-organizations`.

Saved replies

Saved replies, shown in Figure 1-9, can be extremely useful for popular OSS. For example, if you're building an extension to an application, a lot of folks may report problems with the application, not with your extension. You can write a saved reply to point folks to where they can provide feedback on the application when they find an error.


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Saved replies
Add a saved reply
Saved reply title
Add a short title to your reply
Write
Preview
AA B i
Add your saved reply
Add saved reply

Saved replies are re-usable text snippets that you can use throughout GitHub comment fields. Saved replies can save you time if you're often typing similar responses. Learn more.

No saved replies yet.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

FIGURE 1-9: The Saved replies settings.

Applications

You can connect three kinds of applications with your GitHub.com account:

- **Installed GitHub apps:** GitHub applications that you are using with your account. One example is GitHub Learning Labs.
- **Authorized GitHub apps:** Applications that you have authorized to access your account. One example is Slack.
- **Authorized OAuth apps:** Applications that you have authenticated with using GitHub credentials. One example is GitHub Desktop.

Developer settings

The last section on the Settings page is Developer settings, which you use only if you're building an application that accesses the GitHub API, which means the application needs to access GitHub data in some way.

Three settings appear in this section:

- **OAuth apps:** Applications you have registered to use the GitHub API.
- **GitHub apps:** Applications that integrate with and extend GitHub.
- **Personal access tokens:** Similar to SSH keys, tokens that allow you to access the GitHub API without requiring authentication.

Discovering Helpful Resources

The GitHub.com help page (`https://help.github.com`) has an extensive list of documents for every feature on GitHub.com. From the top-right avatar menu, you can click the Help link. From there, clicking Contact a Human takes you to the GitHub contact page (`https://github.com/contact`) where you can find the following resources:

- A FAQ
- Links to the help documentation
- Links to developer documentation for help on the GitHub API (`https://developer.github.com`)
- The GitHub Learning Lab for guided GitHub exercises (`https://`

`lab.github.com`)

- The GitHub Community Forum, where you can ask questions and get answers from folks who work at GitHub and other community members (`https://github.community`)
- A slew of other resources about your experiences on GitHub.com

If none of those resources are helpful, you can visit `https://github.com/contact` and fill out a quick form to ask a specific question.

Chapter 2

Setting Up Your Collaborative Coding Environment

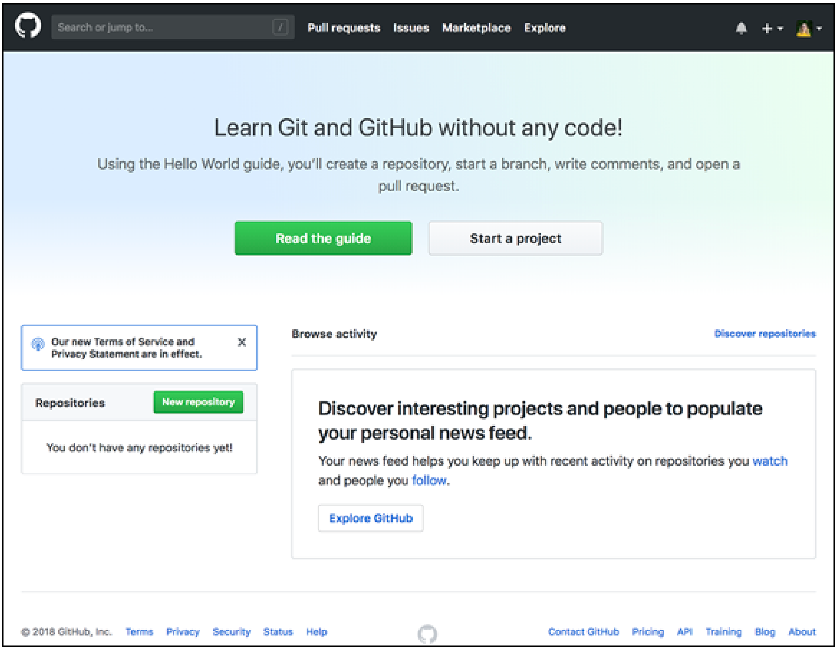
IN THIS CHAPTER

- **Touring GitHub.com**
- **Installing GitHub Desktop**
- **Installing Atom**

GitHub has a lot of features to offer new and returning coders. A good way to get acquainted with all those features is to explore the GitHub.com website. This chapter not only gives you that overview, but also guides you in setting up your local machine so that you can start building.

Exploring GitHub.com

The home page of GitHub.com, shown in Figure 2-1, is a great starting point for many tasks, including starting your own project, learning about a topic, or exploring existing repositories.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Read the guide
Start a project
Browse activity
Discover repositories
Repositories
New repository
Explore GitHub
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Our new Terms of Service and Privacy Statement are in effect.

You don't have any repositories yet!

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch and people you follow.

© 2018 GitHub, Inc.

FIGURE 2-1: The home page of GitHub.com.

The top menu bar, shown in Figure 2-2, is always available to you and is a direct link to the most important functions you need to perform.

- **GitHub home page:** If you click the GitHub logo in the top left of the browser, you return to the home page. Check out the sidebar "Mona Lisa Octocat" for more information on the logo.
- **Search bar:** The search bar on the top menu is pretty snazzy. Not only can you search all of GitHub, but as you start to use the site, it offers suggestions based on your most recent activity. These suggestions make it fast and easy to find the repository you were working on yesterday.
- **Pull requests:** The link to pull requests takes you to a list of all pull requests that you created, were assigned to complete, were mentioned in, or were asked to review. A pull request is a proposed change to the code of a repository. When first starting, you normally don't have anything in this section, but as you start interacting with collaborative repositories, you get an overview of any tasks you may want to

attend to. For more on pull requests, see Chapter 3.

TIP If you click the Pull requests link, you might notice a ProTip, shown in Figure 2-3. The search bar for pull requests gives you several ways to specify a search to get exactly what you're looking for. In fact, an entire page (`https://help.github.com/articles/searching-issues-and-pull-requests`) is dedicated to effective searching. You can find ProTips throughout GitHub.com, so be sure to look out for them.

- **Issues:** The list of issues is almost the same as the list of pull requests. The main difference between an issue and a Pull Request is that an issue is a report of a bug or a feature request. An issue doesn't contain a proposed code change like a pull request does and therefore doesn't require a reviewer.
- **Marketplace:** The marketplace on GitHub is a great place to find applications and tools that can help your collaborative coding workflow. For example, I have used AppVeyor, a continuous integration application, on projects. When you connect AppVeyor to one of your repositories, it continuously runs tests and deploys apps to make sure that every bit of code you're adding won't break what you've already built.
- **Explore:** The Explore link takes you to a list of things you may be interested in (see Figure 2-4). You may find events and opportunities that GitHub hosts or supports. For example, at the time we wrote this book, GitHub just released "The State of the Octoverse," which presents a lot of interesting analytics about code on GitHub — for example, GitHub users made 1.1 billion contributions in 2018!
- **Notifications:** The bell icon leads you to a list of your notifications. See Chapter 1 for how to change your notification settings.
- **Quick pick:** The add-sign icon provides you with a list of quick actions you can take at any time: create a new repository, import a repository from another SCM, create a *gist* (a quick way to share

code, notes, and snippets), or create a new organization.

- **Account menu:** The account menu appears when you click on your avatar. Here, you can get to your profile, repositories, anything you've starred, gists you've created, the help documents, and settings and can sign out.

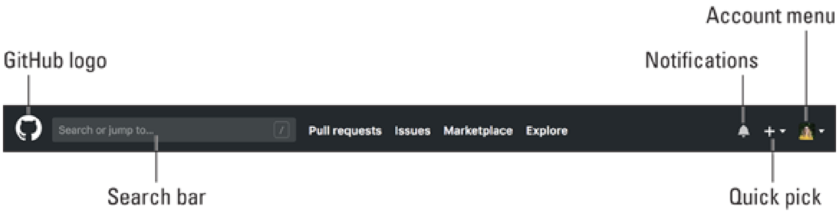
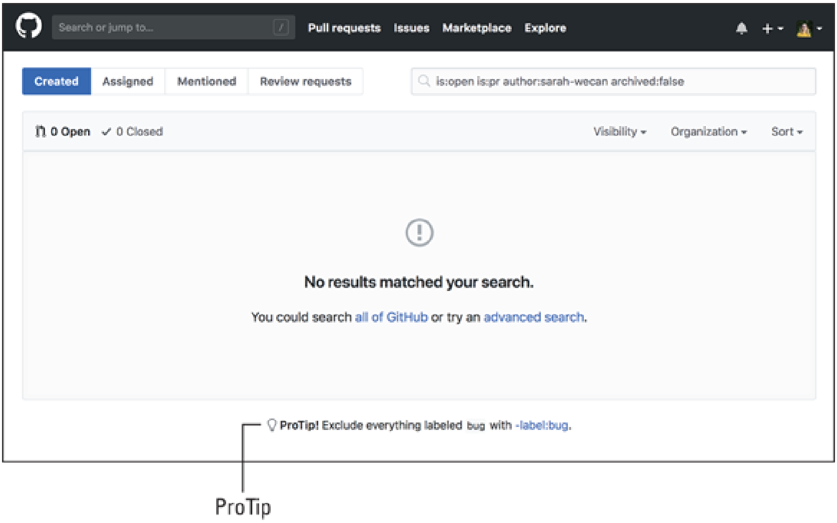

Account menu
GitHub logo
Notifications
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Search bar
Quick pick

FIGURE 2-2: The top menu bar of GitHub.com.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Created
Assigned
Mentioned
Review requests
0 Open
✓ 0 Closed
Visibility
Organization
Sort
ProTip

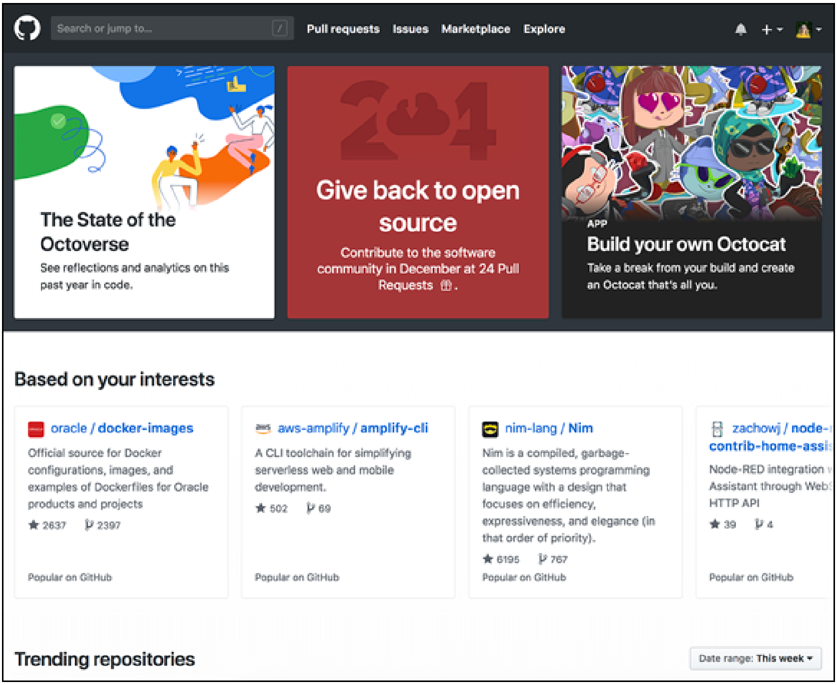
is:open is:pr author:sarah-wecan archived:false

No results matched your search.

You could search all of GitHub or try an advanced search.

ProTip! Exclude everything labeled bug with -label:bug.

FIGURE 2-3: ProTip found on the Pull request page.



The State of the Octoverse

See reflections and analytics on this past year in code.

Give back to open source

Contribute to the software community in December at 24 Pull Requests.

APP

Build your own Octocat

Take a break from your build and create an Octocat that's all you.

Based on your interests

oracle / docker-images

Official source for Docker configurations, images, and examples of Dockerfiles for Oracle products and projects

★ 2637 2397

Popular on GitHub

aws-amplify / amplify-cli

A CLI toolchain for simplifying serverless web and mobile development.

★ 502 69

Popular on GitHub

nim-lang / Nim

Nim is a compiled, garbage-collected systems programming language with a design that focuses on efficiency, expressiveness, and elegance (in that order of priority).

★ 6195 767

Popular on GitHub

zachowj / node-

contrib-home-assis

Node-RED integration

Assistant through Web

HTTP API

★ 39 4

Popular on GitHub

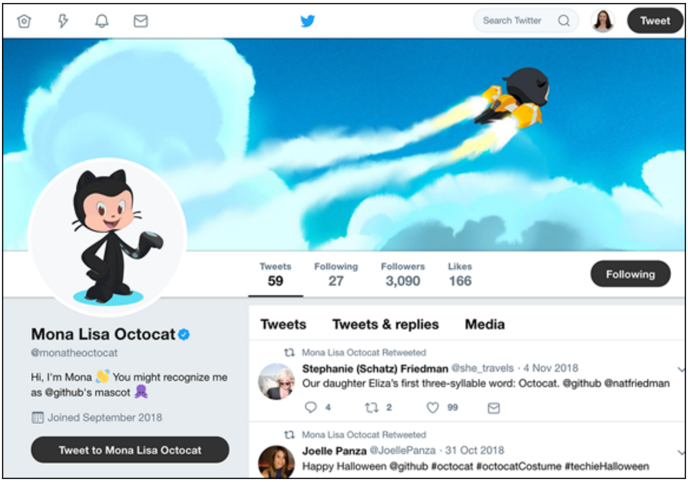
Trending repositories

FIGURE 2-4: Curated list of repositories on GitHub.com.

MONA LISA OCTOCAT

The GitHub logo is a 2D rendition of an *octocat* — a cat with five octopus-like arms. The logo was found on iStock, a website where royalty-free digital images can be purchased. Simon Oxley, the original designer, is also known as the designer for the Twitter bird logo. Cameron McEfee then led the effort around creating an entire Octodex of Octocats, which you can see at `https://octodex.github.com`.

Mona Lisa Octocat also has a verified Twitter account (see figure). You can follow her at `https://twitter.com/monatheoctocat` to get all of the most up-to-date happenings.


Search Twitter
Tweet
Tweets 59
Following 27
Followers 3,090
Likes 166
Following
Tweets
Tweets & replies
Media
Tweet to Mona Lisa Octocat
4
2
99

Mona Lisa Octocat
@monatheoctocat
Hi, I'm Mona 👋 You might recognize me as @github's mascot 🐙
Joined September 2018

Mona Lisa Octocat Retweeted
Stephanie (Schatz) Friedman @she_travels · 4 Nov 2018
Our daughter Eliza's first three-syllable word: Octocat. @github @natfriedman

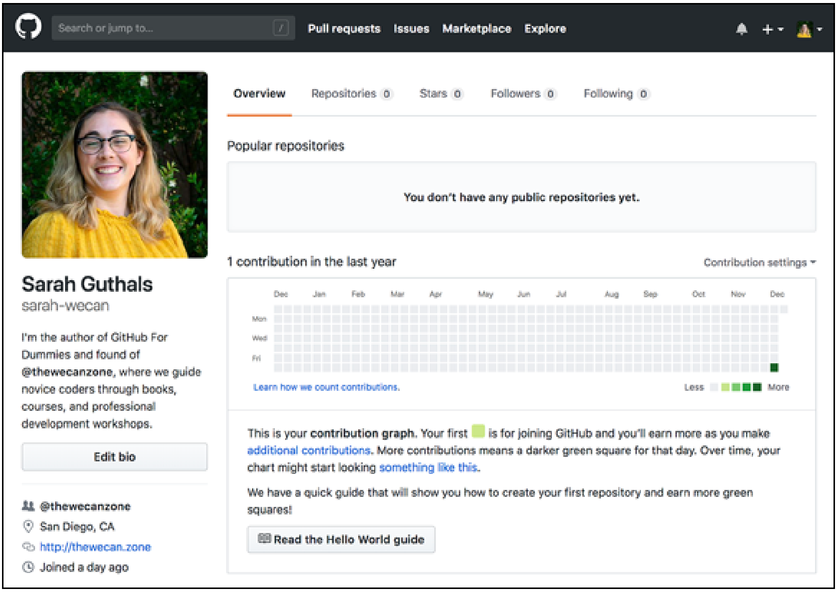
Mona Lisa Octocat Retweeted
Joelle Panza @JoellePanza · 31 Oct 2018
Happy Halloween @github #octocat #octocatCostume #techieHalloween

One of the most popular things Mona has release is the “Build Your Own Octocat” app, which you can find at `https://myoctocat.com/build-your-octocat`. I created one and found my inner super woman!

To discover the full history of the Octocat, visit `http://cameronmcefee.com/work/the-octocat`.

Understanding Your Profile

Your profile is a public view of, essentially, your portfolio. To view your profile, click your avatar and then choose Your profile from the menu that appears. You can see my profile in Figure 2-5.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Overview
Repositories 0
Stars 0
Followers 0
Following 0
Popular repositories
Contribution settings
Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Mon
Wed
Fri
Learn how we count contributions.
Less
More
Edit bio
Read the Hello World guide

Sarah Guthals
sarah-wecan

I'm the author of GitHub For Dummies and found of **@thewecanzone**, where we guide novice coders through books, courses, and professional development workshops.

@thewecanzone
San Diego, CA
http://thewecan.zone
Joined a day ago

You don't have any public repositories yet.

1 contribution in the last year

This is your **contribution graph**. Your first 🟩 is for joining GitHub and you'll earn more as you make additional contributions. More contributions means a darker green square for that day. Over time, your chart might start looking something like this.

We have a quick guide that will show you how to create your first repository and earn more green squares!

FIGURE 2-5: My profile.

The top menu bar of your profile offers quick links to your repositories, things you've starred, your followers, and anyone that you follow. Below that are repositories that you're often visiting and your contribution graph. The *contribution graph* tracks how much code you've written per day. You can choose to include contributions to private repositories and an activity overview, which is a new feature.

WARNING The amount and frequency at which you write code is not the bar by which software development is measured. It is true that the more you practice, the better you will be, but the practice you do must be deliberate. Making random code changes every single day without challenging yourself, or giving yourself time to think, design, and plan the code you want to write, is much worse than missing a white square.

Getting to Know GitHub Desktop

GitHub Desktop is a free, open source application that makes it easier for Mac and Windows users alike to manage repositories and GitHub connections on their local computer.

The fact that Desktop is open source means that you can follow the development of new features, connect with the developers right on the actual repository where the application is being built, and even choose to add features you're interested in having.

You can find the repository at `https://github.com/desktop/desktop`.

To install the GitHub Desktop on your computer:

1. **Go to `https://desktop.github.com` and click Download for the platform you're using.**

   REMEMBER This book is written with examples using Google Chrome and a Mac. GitHub Desktop works on a Windows PC as well because it's built using Electron, which allows it to work on both operating systems.

2. **After the download finishes, click the file that was downloaded.**

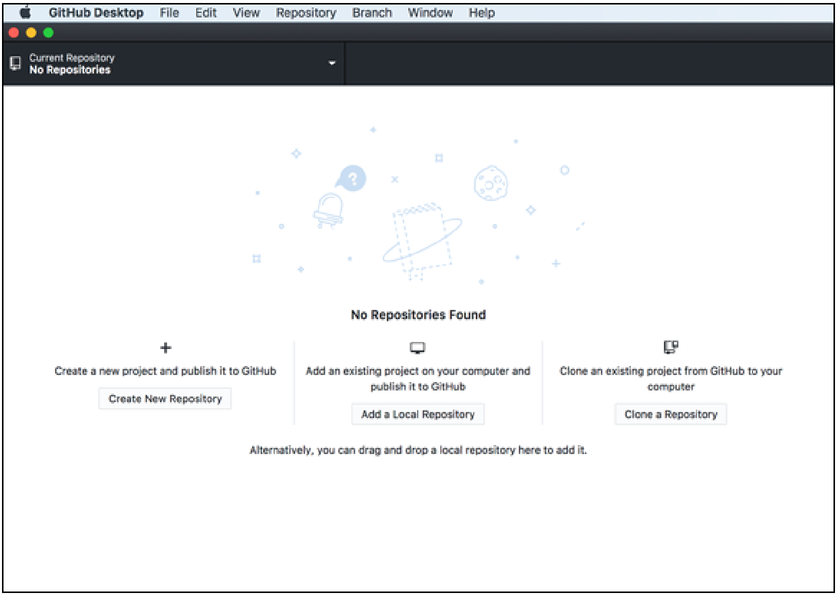
   The file automatically unzips.

   On Mac, the GitHub Desktop application appears in your Downloads folder, next to the zip file. On Windows, the application immediately opens after you unzip the file.

3. **On Mac, drag the purple GitHub Desktop application into your Applications folder.**

4. **On Mac, go to your Applications folder and double-click the GitHub Desktop icon.**

   The application opens, shown in Figure 2-6.


GitHub Desktop
File
Edit
View
Repository
Branch
Window
Help
Current Repository
No Repositories
Create New Repository
Add a Local Repository
Clone a Repository

No Repositories Found

Create a new project and publish it to GitHub

Add an existing project on your computer and publish it to GitHub

Clone an existing project from GitHub to your computer

Alternatively, you can drag and drop a local repository here to add it.

FIGURE 2-6: The GitHub Desktop application default view.

WARNING You may get an alert that you are trying to open an application that was downloaded from the Internet. Click "Open" if this alert appears.

Setting up GitHub Desktop

Before you can use GitHub Desktop effectively, you have to do a few things to connect it to your GitHub.com account. If you do not yet have a GitHub.com account, go to Chapter 1. If you already have a GitHub.com account and have already downloaded GitHub Desktop, you can set up GitHub Desktop with the following steps:

1. **Open the GitHub Desktop application.**
2. **Choose File ⇒ Preferences.**
3. **On the Accounts tab, click Sign In on the GitHub.com row.**

The Sign In dialog box, shown in Figure 2-7, appears.

4. **Type your username and password and click the Sign in button or click Sign in using your browser.**

   When you click Sign in, all the dialog boxes close.

5. **Repeat Step 1 to re-open the preferences.**

   Your account with your avatar, name, and GitHub username appears under the GitHub.com row, confirming that you have successfully logged in.

6. **Click on the Git tab.**

   The information has been auto-filled for you.

7. **On the Appearance tab, choose Light or Dark.**

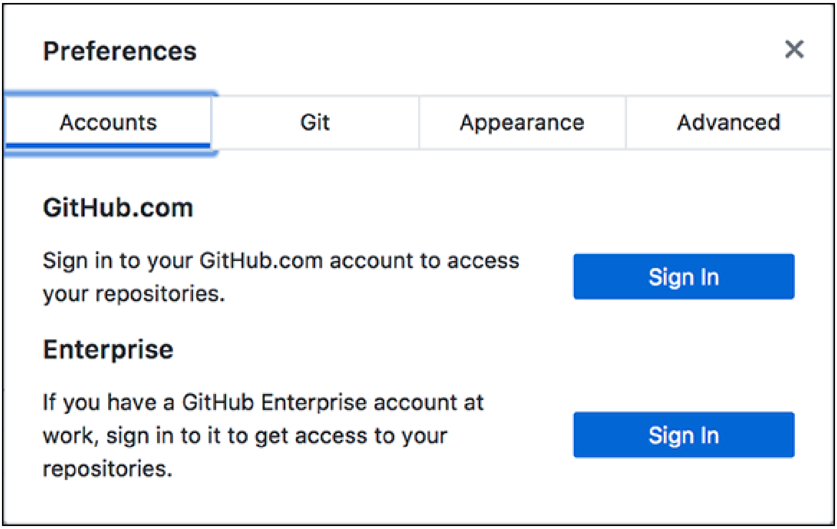
   Screenshots in this book are in Light mode.

8. **Set other preferences, such as the Editor and usage data, on the Advanced tab.**

   This book uses Atom and Visual Studio Code as example editors, but you can select whichever editor you prefer.

   I recommend agreeing to send usage data, which is checked by default. By leaving it checked, you help the GitHub Desktop development team understand how all users use the application and therefore make it better.

   If you do not have a GitHub repository on your computer yet, you can stop the setup here. If you do have a repository, see Chapter 4.



GitHub.com

Sign in to your GitHub.com account to access your repositories.

Enterprise

If you have a GitHub Enterprise account at work, sign in to it to get access to your repositories.

FIGURE 2-7: The Sign in dialog box for the GitHub Desktop application.

REMEMBER While a team of folks at GitHub predominately does the development of GitHub Desktop, a part of their role is to support community members who want to contribute to the project. Don't hesitate to reach out to the team on its repository at `https://github.com/desktop/desktop`.

Introducing Atom

Atom is a free, open-source editor. Just like GitHub Desktop, Atom is built on Electron, making it work on Mac or Windows PC. Atom is *extensible,* meaning you can add your own features to it.

You can take a look at what the Atom team at GitHub is working on by visiting the repository; `https://github.com/atom/atom`.

Atom is a lightweight editor that shouldn't take long to install. To install Atom, go to `https://atom.io` and click Download.

Just like with GitHub Desktop in the previous section, when Atom finishes downloading, click to unzip the file. On Mac, the Atom application appears in your Downloads folder. Drag the Atom application into your applications folder. On Windows or Mac, double-click the application to open it. When you do, you should see what is shown in Figure 2-8.

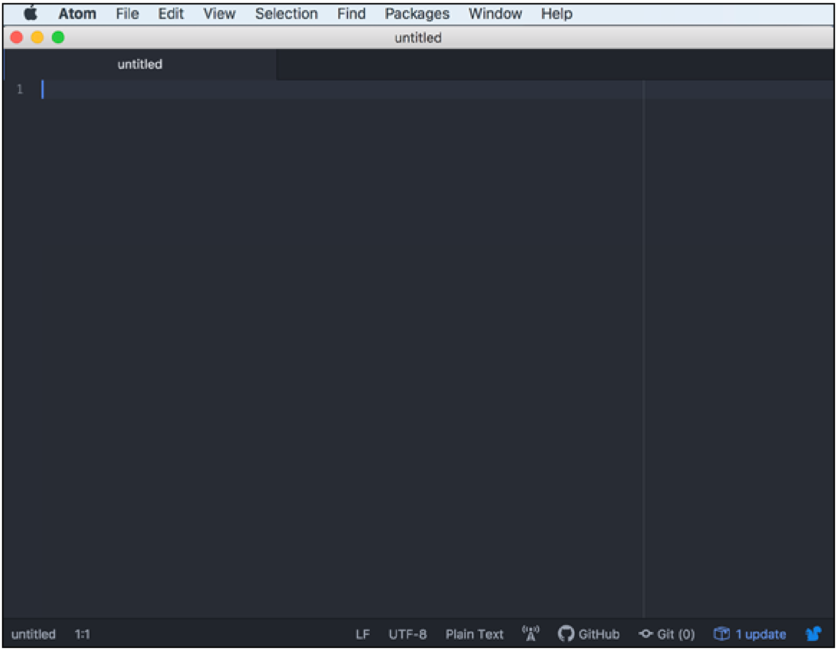

Atom File Edit View Selection Find Packages Window Help
untitled
untitled
1
untitled 1:1
LF UTF-8 Plain Text
GitHub
Git (0)
1 update

FIGURE 2-8: The Atom application default view.

WARNING You may get an alert that you're trying to open an application that was downloaded from the Internet. Click Open if this alert appears.

Here are a few things that you should know about Atom:

- **Updates:** Each time you start Atom, make sure you check the bottom- right corner to see whether any updates need to happen so that you keep your software as current as possible. When you click the Update link, the Settings tab opens, and the packages appear on the screen. In Figure 2-9, you can see that I had previously installed the package `teletype`, which requires an update. After clicking Update to 0.13.3 or Update All if I have multiple packages that need updating, we restart Atom.

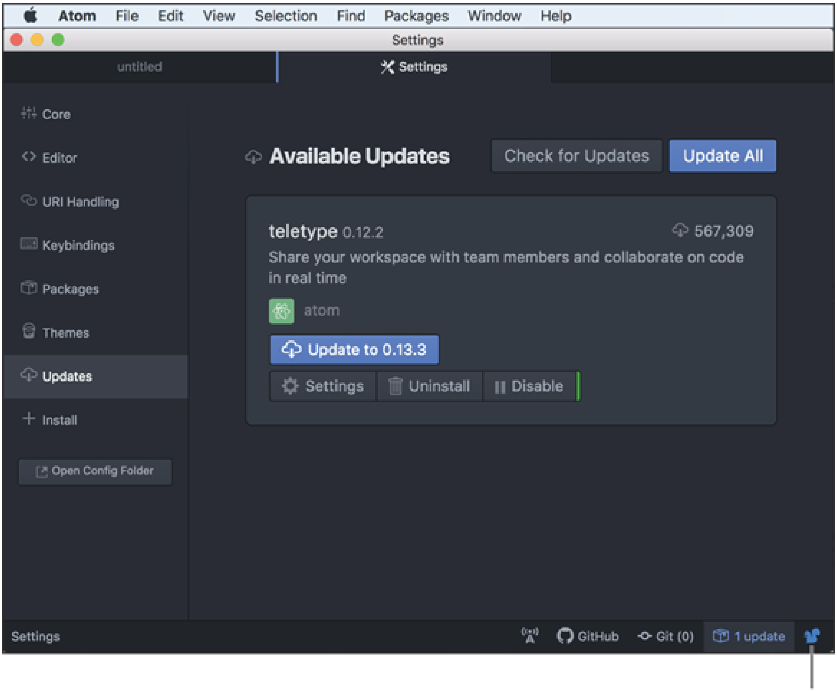
TIP You can click the Squirrel icon to go to the Atom About page to ensure that your Atom editor is also kept up-to-date.

- **Git and GitHub tabs:** At the bottom, right, next to where the updates are, you can easily open the Git and GitHub tabs by choosing Packages ⇒ GitHub ⇒ Toggle Git/GitHub Tab.

TIP You may notice that some menu items throughout Atom have keyboard bindings, such as the Git and GitHub tabs in Figure 2-10. *Keyboard bindings* are combinations of keys you can press on your keyboard to make something happen in a specific application. You probably already know some of these. For example, when you're browsing the Internet, you can type ⌘-T on a Mac or Ctrl+T on Windows to open a new tab. Finding ways to become more efficient in your coding, such as by using keyboard bindings, can be an effective strategy for you as you become more expert in your coding journey.

- **Preferences:** You can specify a lot of preferences for Atom. I don't explain each one in this book, but I encourage you to browse them and really set up Atom to make it exactly right for you. You can find the preferences by choosing Atom ⇒ Preferences (see Figure 2-11).

- **Packages:** You can install more than 8,000 Atom packages to make Atom most effective for you. You can find them under preferences or at `https://atom.io/packages`. Going through each of these packages is beyond the scope of this book, but I encourage you to explore some and search them if you're ever feeling limited by your Atom experience.


Atom
File
Edit
View
Selection
Find
Packages
Window
Help
Settings
untitled
Settings
Core
Editor
URI Handling
Keybindings
Packages
Themes
Updates
Install
Open Config Folder
Available Updates
Check for Updates
Update All
teletype 0.12.2
567,309
atom
Update to 0.13.3
Settings
Uninstall
Disable
Settings
GitHub
Git (0)
1 update

Share your workspace with team members and collaborate on code in real time

Update

FIGURE 2-9: the Squirrel icon takes you to the About page where you can update Atom.

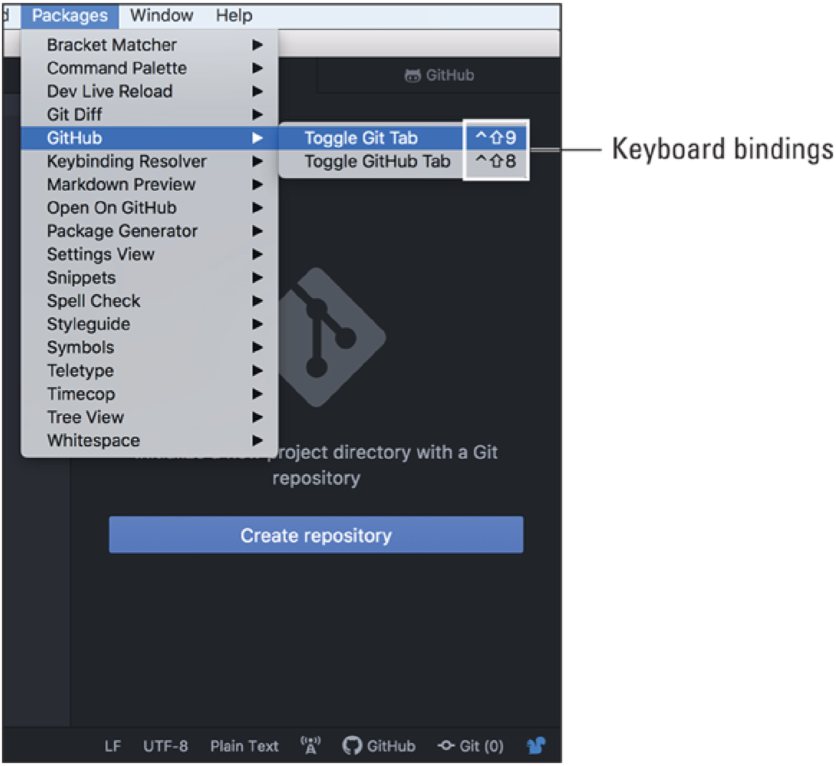
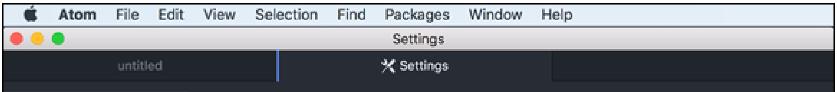
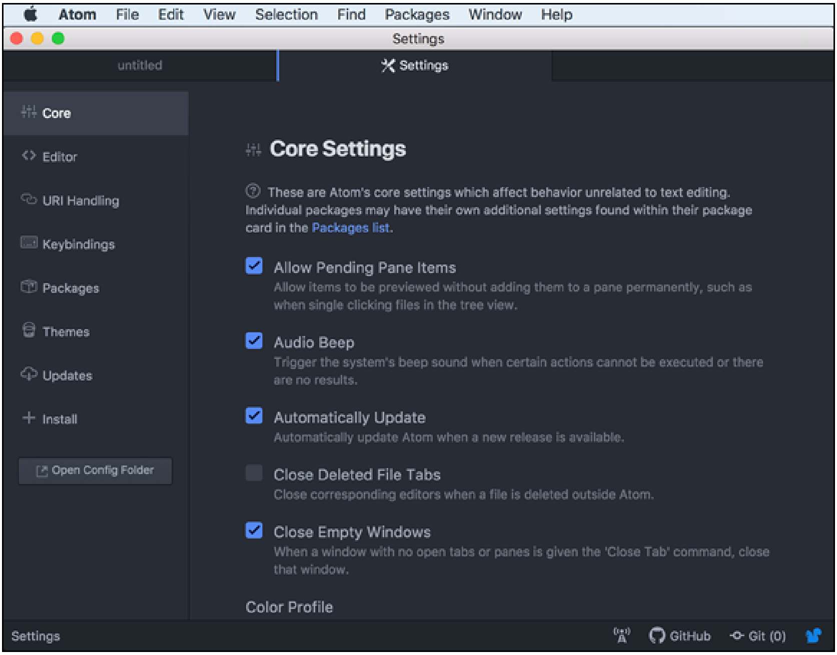

Packages
Window
Help
Bracket Matcher
Command Palette
Dev Live Reload
Git Diff
GitHub
Keybinding Resolver
Markdown Preview
Open On GitHub
Package Generator
Settings View
Snippets
Spell Check
Styleguide
Symbols
Teletype
Timecop
Tree View
Whitespace
Toggle Git Tab ^⇧9
Toggle GitHub Tab ^⇧8
Keyboard bindings
GitHub
roject directory with a Git repository
Create repository
LF
UTF-8
Plain Text
GitHub
Git (0)

FIGURE 2-10: Keyboard bindings for toggling the Git and GitHub tabs.


Atom
File
Edit
View
Selection
Find
Packages
Window
Help
Settings
untitled
Settings


Atom
File
Edit
View
Selection
Find
Packages
Window
Help
Settings
untitled
Settings
Core
Editor
URI Handling
Keybindings
Packages
Themes
Updates
Install
Open Config Folder
Settings
GitHub
Git (0)

Core Settings

These are Atom's core settings which affect behavior unrelated to text editing. Individual packages may have their own additional settings found within their package card in the Packages list.

☑ Allow Pending Pane Items
Allow items to be previewed without adding them to a pane permanently, such as when single clicking files in the tree view.

☑ Audio Beep
Trigger the system's beep sound when certain actions cannot be executed or there are no results.

☑ Automatically Update
Automatically update Atom when a new release is available.

☐ Close Deleted File Tabs
Close corresponding editors when a file is deleted outside Atom.

☑ Close Empty Windows
When a window with no open tabs or panes is given the 'Close Tab' command, close that window.

Color Profile

FIGURE 2-11: The Atom Settings page.

REMEMBER If you're looking for resources outside of this book, check out the Atom documentation at `https://atom.io/docs` and the Atom Flight Manual at `https://flight-manual.atom.io`. Both of these resources can help guide you through any problem you may encounter. If they don't, you can reach out to the developers who work on Atom by visiting the open source repository at `https://github.com/atom/atom`.

Part 2

Starting Your First Solo Project

IN THIS PART …

Create a GitHub repository to store code.

Explore GitHub issues, pull requests, and project boards to manage your code.

Create a GitHub Pages website for a project.

Sync your remote GitHub project with your local coding environment.

Create a GitHubb repository specfically for creating a personal website.

Add a blog to a GitHub Pages website.

Chapter 3

Introducing GitHub Repositories

IN THIS CHAPTER

- **Touring a repository?**
- **Creating a Hello World repository**
- **Exploring repository issues, pull requests, and project boards**

Almost everything on GitHub.com revolves around a repository.

In this chapter, you find out how you can set up a repository, interact with it, and create project boards and issues.

Setting Up a Repository

A GitHub repository is a folder with all the files needed for your project, including the files that track all the versions of your project so that you can revert back if you make a mistake. A repository on GitHub also tracks who can collaborate and how.

To get a better understanding of what a repository is and how it is structured, you need to create your first GitHub repo:

1. **Go to the home page of GitHub.com by clicking the Octocat.**

   A list of your repositories appears on the bottom left side of the screen.

2. **Click the green New Repository button.**

The Create a New Repository dialog page, shown in Figure 3-1, opens.

3. **Type the name of your repository in the Repository name text box.**

   I named my repository `HelloWorld`.

4. **Type a short description in the Description text box.**
5. **Select the Public radio button.**
6. **Click the Initialize this repository with a README check box.**

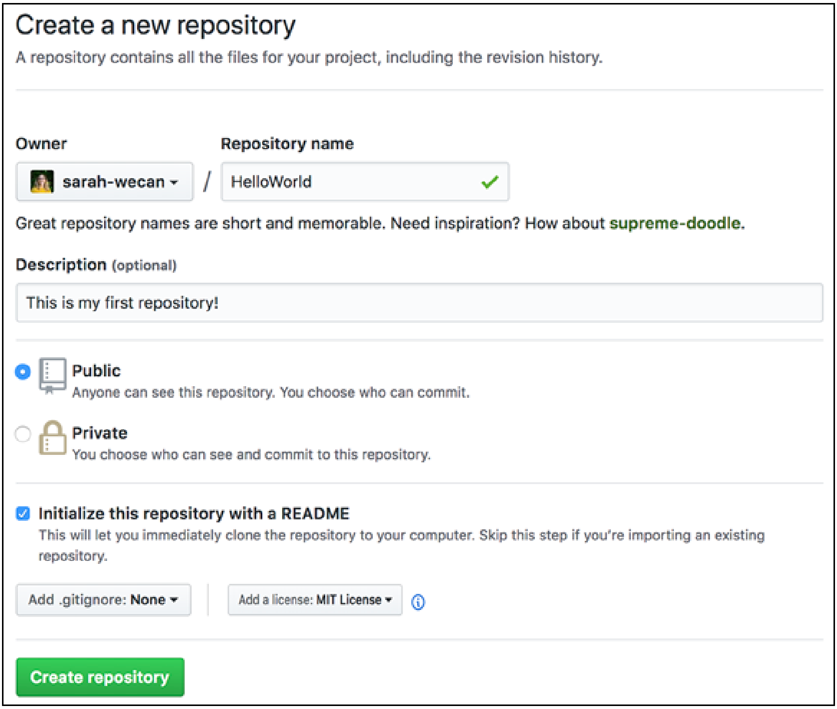
   You do not need to add a `.gitignore`.

7. **Choose a license from the Add a license drop-down list.**

   TIP If you're interested in finding out more information about licenses, see the nearby "Software licenses" sidebar.

8. **Click Create Repository.**

   The home page of your repository appears. It should look similar to the one I created, which is shown in Figure 3-2. Notice that a markdown file — README.md — is already in the repository. *Markdown* is a lightweight markup language used to style the words that you write with a plain text syntax. You can make words bold, turn them into headers, and even create a table for data.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: sarah-wecan

Repository name: HelloWorld ✓

Great repository names are short and memorable. Need inspiration? How about **supreme-doodle**.

Description (optional): This is my first repository!

● **Public** Anyone can see this repository. You choose who can commit.

○ **Private** You choose who can see and commit to this repository.

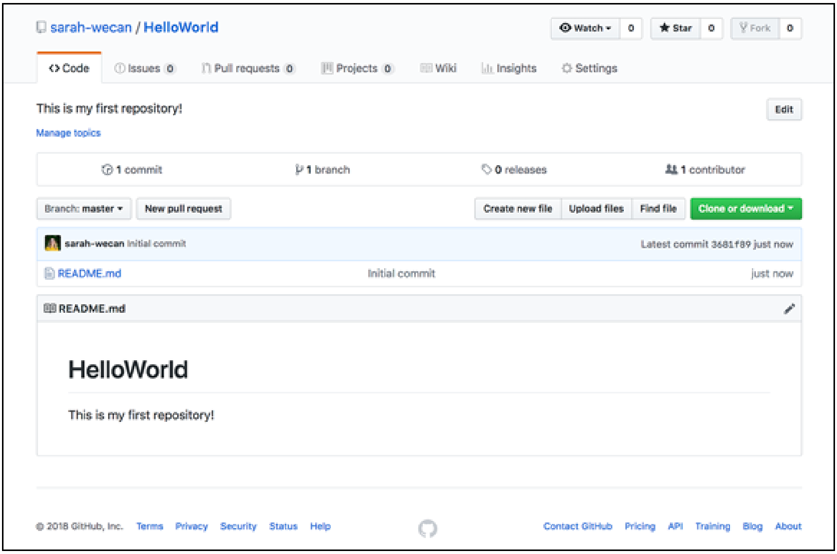
☑ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **MIT License**

Create repository

FIGURE 3-1: The page to create a new repository.



sarah-wecan / HelloWorld

Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Edit
Manage topics

This is my first repository!

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

sarah-wecan Initial commit | Latest commit 3681f89 just now

README.md | Initial commit | just now

README.md

HelloWorld

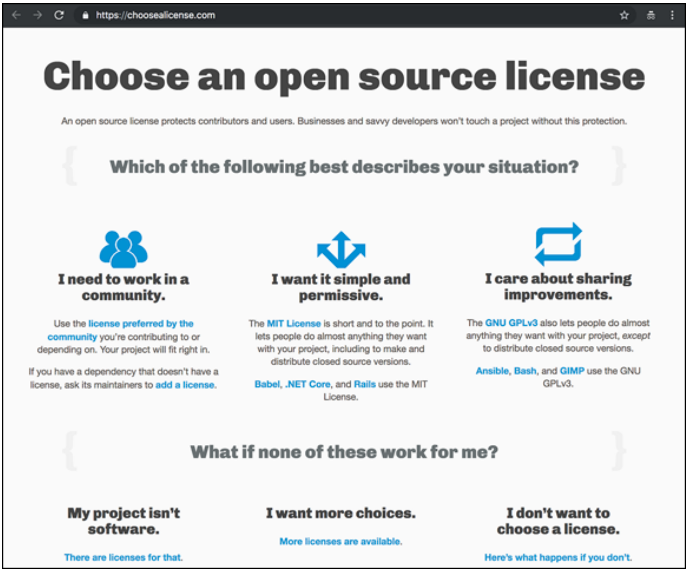
This is my first repository!

© 2018 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

FIGURE 3-2: The home page of my HelloWorld repository.

SOFTWARE LICENSES

Software licenses are a really important part to collaborative coding. Whether you are putting your code up on GitHub.com to share with the world, or contributing to someone else's code, you should know what is allowed and what isn't. When you first create a new repo, you're given the option to attach a license to it, as you see in Figure 3-2. If you click on the question-mark, you will be taken to `https://choosealicense.com/`, which explains the top three most common software license types.



Choose an open source license

An open source license protects contributors and users. Businesses and savvy developers won't touch a project without this protection.

Which of the following best describes your situation?

I need to work in a community.

Use the license preferred by the community you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to add a license.

I want it simple and permissive.

The MIT License is short and to the point. It lets people do almost anything they want with your project, including to make and distribute closed source versions.

Babel, .NET Core, and Rails use the MIT License.

I care about sharing improvements.

The GNU GPLv3 also lets people do almost anything they want with your project, *except* to distribute closed source versions.

Ansible, Bash, and GIMP use the GNU GPLv3.

What if none of these work for me?

My project isn't software.

There are licenses for that.

I want more choices.

More licenses are available.

I don't want to choose a license.

Here's what happens if you don't.

The two main license types used for collaborative, public software are the MIT License and the GNU General Public License v3.0 License. The MIT License allows people to do almost anything with the code they find in your project. This includes being able to take a copy, make changes (or not) and distribute it as a closed source version. That means that they can distribute the software as an application without giving *their* users access to the code. The GNU General Public License v3.0 License also gives folks access to copy, modify, and contribute to your project, but if they want to distribute their version, it must be public and open. The repositories in this book use the MIT License, but you can choose what you want to do with your projects.

In Chapters 4 and 5, you can create a website for yourself. This website can link back to your repository.

Exploring Your Repository

A repository has a lot going on, even when it's as simple as the HelloWorld one that I created in the preceding section. The following sections walk you through an overview of everything on the repository.

Top information

At the top of the repository is the username of the author and title of the repository. When you fork a repository, you see the original author underneath for a quick link. To *fork* a repository is to make a copy of it, where the changes you make to your copy can be suggested to the original author. See Chapter 6 for a deep dive into forking a repository.

To the right of your username are three buttons:

- **Watch:** You can choose what kind of notifications you want to receive based on the type of activity happening on this repo.
- **Star:** Starring can help you quickly navigate to certain repositories, as well as give GitHub insight into things you're interested in so that recommendations will be more accurate for you. To access your starred repositories, just click your avatar on the top right of GitHub.com and choose Your Stars.
- **Fork:** If you are not the author of the repository, then you have the option to fork it. Chapter 6 goes into more detail on forks.

WARNING We highly recommend choosing either Not watching or Releases only for the majority of repositories so that you only get notifications when you're specifically mentioned or actively participating in a discussion on an issue or a pull request. Otherwise, your inbox will be filled with emails about every single action taken on the repository, which will get out of hand very quickly. If you notice this happening, go to `https://github.com/watching` and unwatch all or some of the repositories you're watching with a quick click.

Tabs

Seven tabs appear across the top of your repo. Each tab provides different features for the repo:

- **Code:** The Code tab is where you can find all your code and browse folders and files. You can click a file to view its contents or click the pencil icon to modify the file, right in your Internet browser. (See the upcoming "Code tab" section for more details.)

- **Issues:** Issues are a really neat feature for repos. Issues can help you track things you want to still make, problems you're having, or suggestions for other people. You discover how to create issues in the upcoming section "Using Issues and Project Boards."

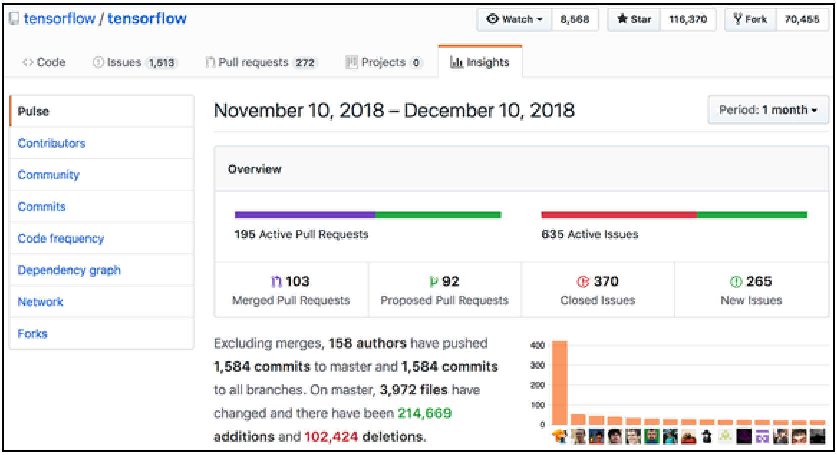
- **Pull requests:** Pull requests, also referred to as PRs, are similar to issues in that they have a title and a description, but they also have code changes that you're requesting to be pulled into the main branch. The safest way to contribute code is to create a new branch, make your code changes on that branch, and then request for that branch to be merged with the master branch. A PR gives you an interface for merging the two branches, showing you the diff between the files you modified and the ones that are on the master branch and giving you a place to have a conversation with collaborators on whether the code should be merged or changes should be made first. For more information on branching, see Chapter 1.

- **Projects:** You may already be familiar with project boards like Trello or Kanboard. GitHub has project boards linked directly with your repo. The best part is that the cards on a GitHub project board can be directly related to issues or PRs and can automatically move when something happens. For more on project boards, see the section "Using Issues and Project Boards," later in this chapter.

- **Wiki:** Wikis are a great place to store documentation, project status, and roadmaps for your project. It's a great go-to place for collaborators to see what is going on and where they can jump in to help!

- **Insights:** The Insights tab, shown in Figure 3-3, gives you an overview of all collaborators and actions happening on the repo. It's really neat to see this tab on popular open source projects. For example, TensorFlow has had 158 contributors in the last month!

- **Settings:** The Settings tab is visible only if you have the right permissions on the repository. In this tab, you can decide who has access to what and how collaborators should collaborate. You can also integrate apps that tell you how much of your code is covered with tests.


tensorflow / tensorflow
Watch 8,568
Star 116,370
Fork 70,455
Code
Issues 1,513
Pull requests 272
Projects 0
Insights
Pulse
Contributors
Community
Commits
Code frequency
Dependency graph
Network
Forks
Period: 1 month
Overview
400
300
200
100
0

November 10, 2018 – December 10, 2018

195 Active Pull Requests

635 Active Issues

103 Merged Pull Requests

92 Proposed Pull Requests

370 Closed Issues

265 New Issues

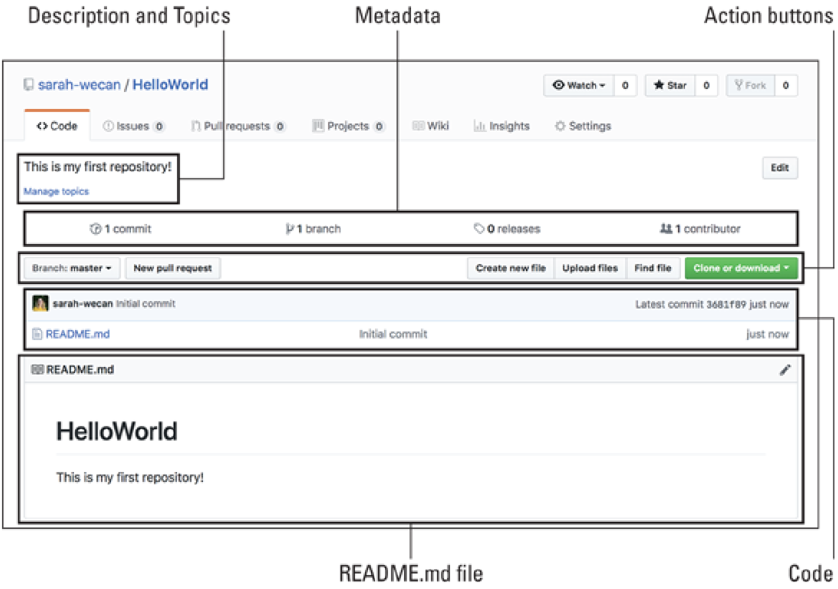
Excluding merges, **158 authors** have pushed **1,584 commits** to master and **1,584 commits** to all branches. On master, **3,972 files** have changed and there have been **214,669 additions** and **102,424 deletions**.

FIGURE 3-3: The Insights tab.

Code tab

The Code tab, shown in Figure 3-4, has a lot of additional important metadata about your repo that will come in useful in future development:

- **Description and topics:** At the top of the Code tab is a description and a place where you can put in topics to make your repository more discoverable. Adding topics is particularly important if you want to attract other coders to help you build your software.
- **Metadata:** This bar contains information and links to the number of commits, branches, releases, and contributors for the repo.
- **Action buttons:** On the left side of the repo is a drop-down menu where you can change which branch you're looking at or browse the files for a particular branch. The New pull request button allows you to quickly create a pull request. The best way to create a pull request is to switch to another branch, make some changes, and then click New pull request. On the right side are three buttons related to files: Create new file, Upload files, and Find file. Finally, you can click on the green Clone or download drop-down list to clone or download the code to your local machine (see Chapter 4).
- **Code:** At the bottom of the Code tab is a list of all the code in this repo. If a `README.md` file appears in this list, then the file shows up below the list. For any file, you can click the filename to go to a page where you can see the file and edit it if you want.


Description and Topics
Metadata
Action buttons
sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
<> Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Manage topics
Edit
1 commit
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
sarah-wecan Initial commit
Latest commit 3681f89 just now
README.md
Initial commit
just now
README.md
README.md file
Code

This is my first repository!

HelloWorld

This is my first repository!

FIGURE 3-4: The Code tab.

Modifying README.md

I highly recommend that every project, whether public or private, have a README.md file at the top level. This file is often the starting point for anyone who wants to contribute to the code.

The README.md file will often have the following sections:

- Project title and description
- Prerequisites for getting the project running on your local machine
- Instructions on installing the project (and any dependencies)
- Instructions on running tests to make sure that you haven't broken anything
- Instructions on deploying the project
- An overview of dependencies
- A link to the guide on how to contribute to the project, including a code of conduct
- The main authors or maintainers of the project

- A link to the license
- Any additional acknowledgements

TIP PurpleBooth on GitHub has created a great template for a `README.md` at `https://gist.github.com/PurpleBooth/109311bb0361f32d87a2`.

REMEMBER GitHub promotes a culture of sharing and open software development. In the sharing, it is important that each person acknowledge where they drew inspiration and what pieces went into helping them create what they have created. Software development rarely happens alone and, at this point, is always built on someone else's work. Though you don't have to specifically acknowledge the work that Grace Hopper, a well-known computer scientist who created the first compiler and English programming language, did to promote high-level programming languages so that you're not all writing in assembly anymore, you should always recognize that it's a large, timeless community working toward building, creating, and pushing the boundaries of what you think is possible today.

For simpler projects, a README.md file can also be the front page to your project. In this case, the project is a description of you, and the README.md file is essentially the entire project because it will contain the information about you!

Follow these steps to modify your README with a headshot and a short description about things you love:

1. **Go to the Code tab of your repository and click the Upload files button.**

   Steps 2 through 6 guide you through the page shown in Figure 3-5.

2. **Find a headshot of yourself (or any picture that**

you want to be on your README) and upload it using the drag box.

Alternatively, you can click the Choose your files link and browse your files to find your photo.

3. **Type a title in the text box Commit changes.**

   In this example, I added the title “Adding a Headshot.”

4. **Select Create a new branch for this commit and start a pull request.**

5. **Type a name for the branch, or you can leave the default name for the branch.**

   I left my default name, which is `sarah-wecan-patch-1`.

6. **Then click the Propose changes button.**

   The Open a pull request page appears with your one commit to add a new headshot to your repo (see Figure 3-6).

7. **Add a short description to explain the change and then click the Create pull request button.**

   The Pull request tab displays your pull request with a title, number, status, description, list of commits, and a check to see whether it can be merged with master, shown in Figure 3-7.

8. **Click the Code tab to go back to the code in your repo.**

9. **Click the branch drop-down list and switch to the branch you created in Step 5.**

   This step allows you to see your project as it looks in that branch. The changes you make added to that branch as you make them and therefore the pull request that you created in Step 6. That way, you can add your headshot and make all the changes to the

README at once.

You see the new branch you created. Next to the branch is a link to the pull request you created. This branch is associated with that pull request, so you can kind of think of them as the same things. You also see the picture that you uploaded in your list of files.

10. **Because your README file is still showing, click the little pencil so that you can change what the README says.**

    If it exists, a README file always appears below the list of files.

11. **Using Markdown, write a little about yourself, including your career passions and some hobbies you enjoy.**

    Figure 3-8 shows you an example of the type of information you may want to include.

12. **Click the Preview changes tab above the text.**

    A red line appears to the left of the first two lines to indicate they will be deleted (see Figure 3-9). Everything you wrote after that has a green line to the left of it to indicate that the text will be added to your description.

13. **When you're satisfied with your description, scroll to the bottom of the file editor, add a title to the commit, and commit to the same branch you just created.**

    You see your README.md file in its final state.

14. **Click the `HelloWorld` link at the top where it says `HelloWorld/README.md` to return to your code home page.**

15. **Add your headshot by clicking on the pencil above the README.md file and adding** the following line:

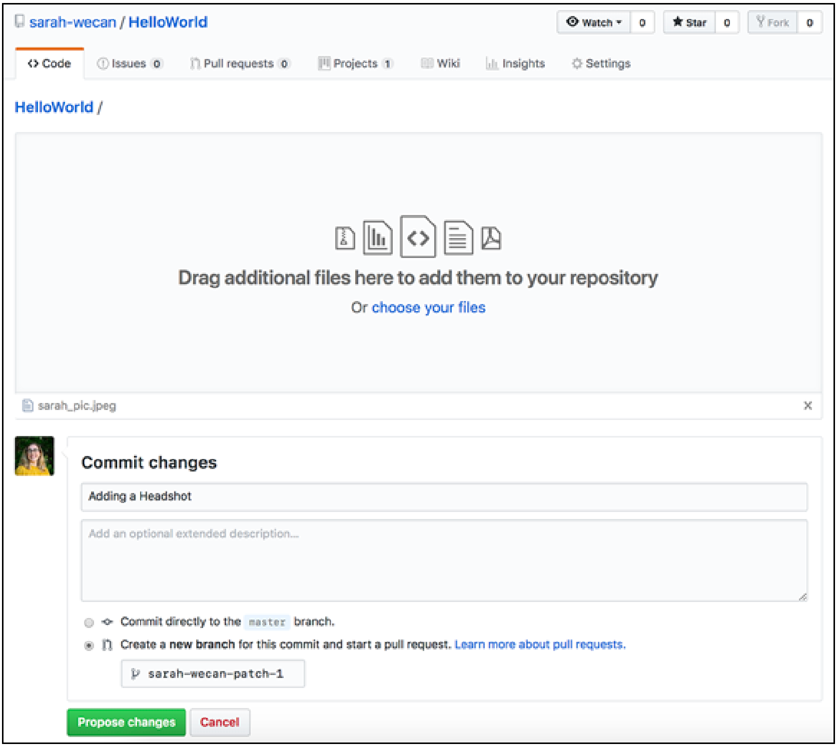
    ```
    ![headshot](sarah_pic.jpeg)
    ```

Make sure that you put the name of your picture in place of `sarah_pic.jpeg`.

16. **Preview the changes.**

    Your headshot now appears. You can see mine in Figure 3-10.

17. **Scroll to the bottom and commit your changes.**


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
HelloWorld /
Drag additional files here to add them to your repository
Or choose your files
sarah_pic.jpeg
Propose changes
Cancel

Commit changes

Adding a Headshot

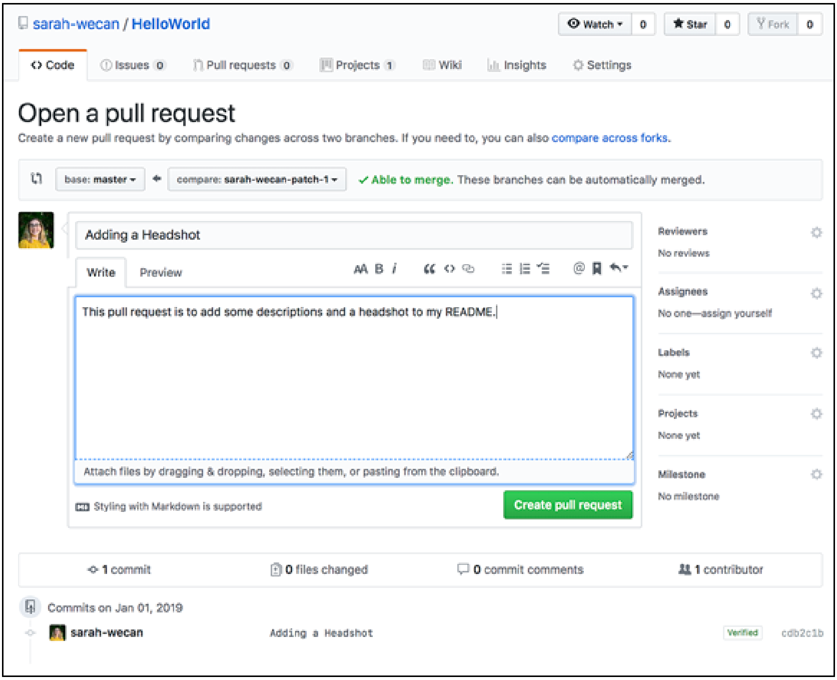
Add an optional extended description...

○ Commit directly to the `master` branch.

● Create a new branch for this commit and start a pull request. Learn more about pull requests.

`sarah-wecan-patch-1`

FIGURE 3-5: The page to add a file to the `HelloWorld` repository.


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
Open a pull request
base: master
compare: sarah-wecan-patch-1
Write
Preview
Styling with Markdown is supported
Create pull request
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
1 commit
0 files changed
0 commit comments
1 contributor
Commits on Jan 01, 2019
sarah-wecan
Adding a Headshot
Verified
cdb2c1b

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

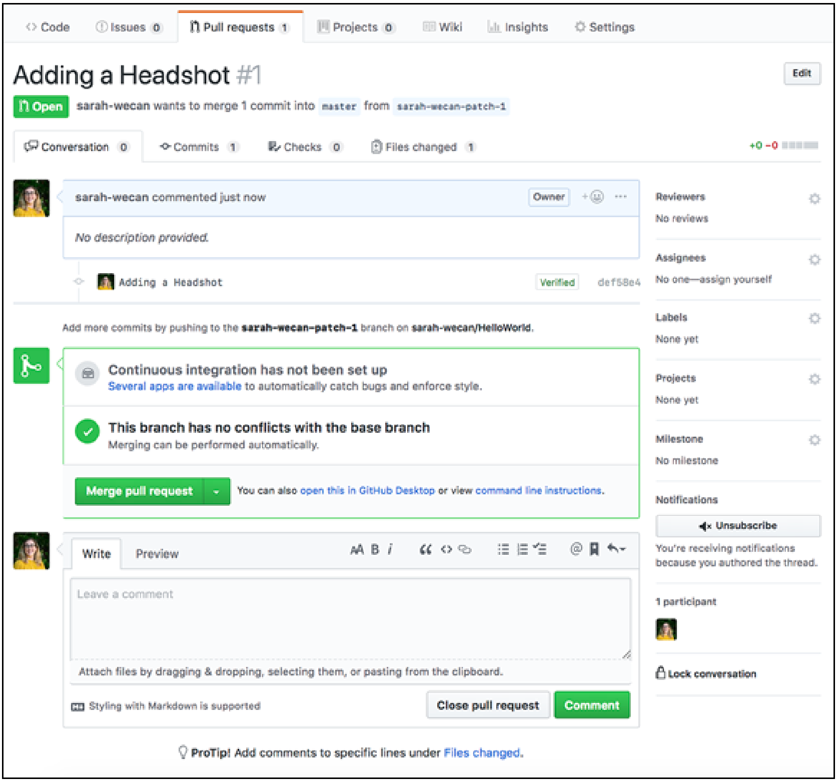
✓ Able to merge. These branches can be automatically merged.

Adding a Headshot

This pull request is to add some descriptions and a headshot to my README.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

FIGURE 3-6: The Open a pull request page.


Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
Settings
Edit
Open
Conversation 0
Commits 1
Checks 0
Files changed 1
+0 −0
Owner
Verified
def58e4
Reviewers
Assignees
Labels
Projects
Milestone
Notifications
Unsubscribe
1 participant
Lock conversation
Merge pull request
Write
Preview
Close pull request
Comment

Adding a Headshot #1

sarah-wecan wants to merge 1 commit into `master` from `sarah-wecan-patch-1`

sarah-wecan commented just now

No description provided.

Adding a Headshot

Add more commits by pushing to the **sarah-wecan-patch-1** branch on **sarah-wecan/HelloWorld**.

Continuous integration has not been set up

Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

ProTip! Add comments to specific lines under Files changed.

No reviews

No one—assign yourself

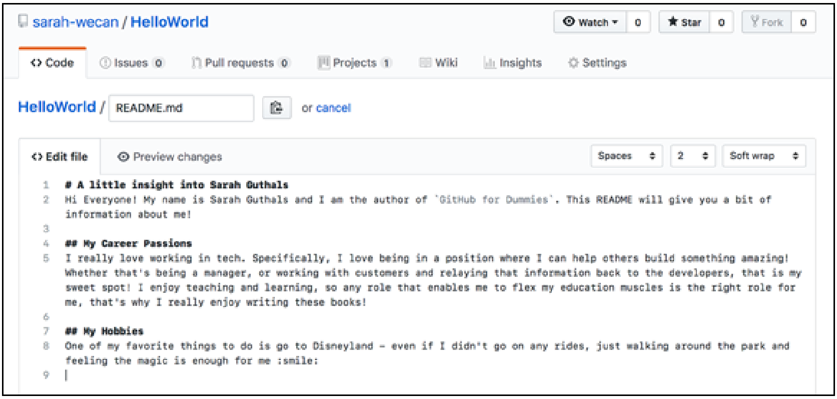
None yet

None yet

No milestone

You're receiving notifications because you authored the thread.

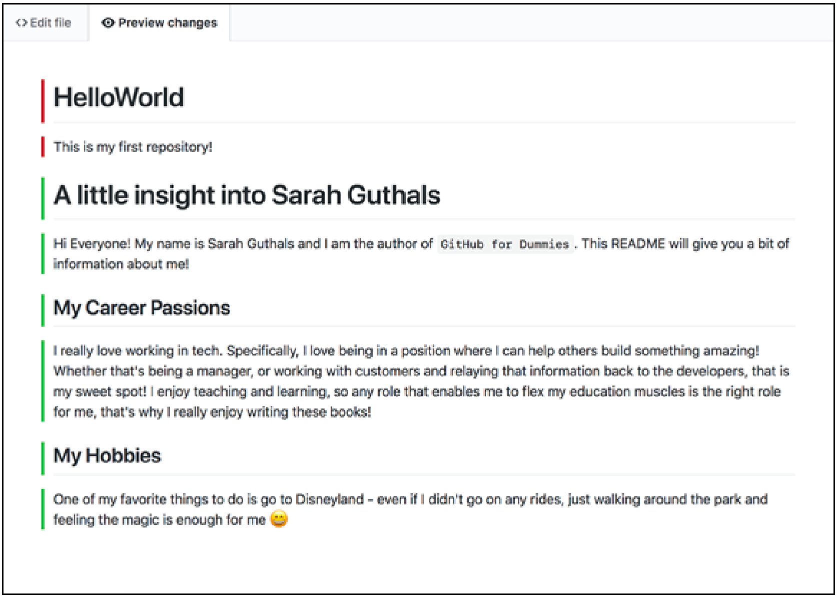
FIGURE 3-7: The pull request description page for an open pull request.


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
HelloWorld /
or cancel
Edit file
Preview changes
Spaces
2
Soft wrap

README.md

```
# A little insight into Sarah Guthals
Hi Everyone! My name is Sarah Guthals and I am the author of `GitHub for Dummies`. This README will give you a bit of information about me!

## My Career Passions
I really love working in tech. Specifically, I love being in a position where I can help others build something amazing! Whether that's being a manager, or working with customers and relaying that information back to the developers, that is my sweet spot! I enjoy teaching and learning, so any role that enables me to flex my education muscles is the right role for me, that's why I really enjoy writing these books!

## My Hobbies
One of my favorite things to do is go to Disneyland - even if I didn't go on any rides, just walking around the park and feeling the magic is enough for me :smile:

```

FIGURE 3-8: The `README.md` file in `edit` mode.



HelloWorld

This is my first repository!

A little insight into Sarah Guthals

Hi Everyone! My name is Sarah Guthals and I am the author of `GitHub for Dummies`. This README will give you a bit of information about me!

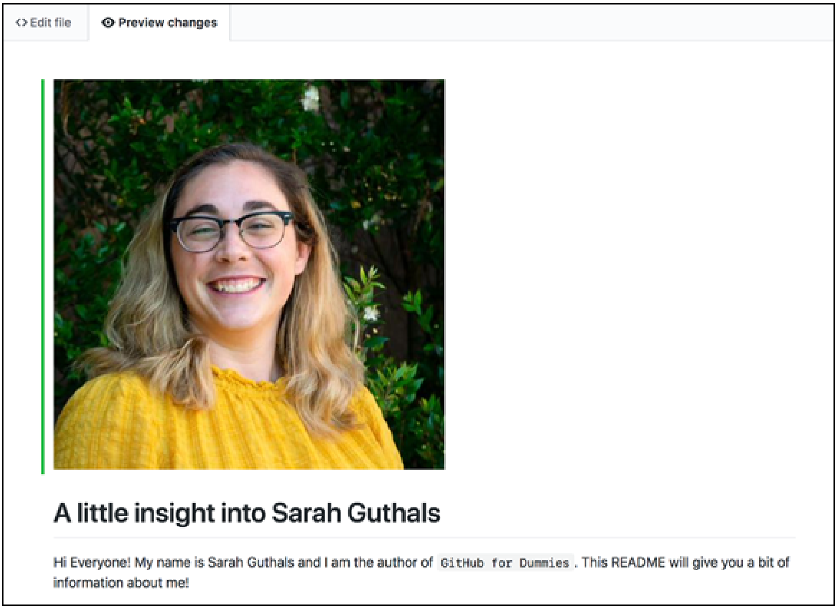
My Career Passions

I really love working in tech. Specifically, I love being in a position where I can help others build something amazing! Whether that's being a manager, or working with customers and relaying that information back to the developers, that is my sweet spot! I enjoy teaching and learning, so any role that enables me to flex my education muscles is the right role for me, that's why I really enjoy writing these books!

My Hobbies

One of my favorite things to do is go to Disneyland - even if I didn't go on any rides, just walking around the park and feeling the magic is enough for me 😀

FIGURE 3-9: The `README.md` file in `diff` mode.



A little insight into Sarah Guthals

Hi Everyone! My name is Sarah Guthals and I am the author of `GitHub for Dummies`. This README will give you a bit of information about me!

FIGURE 3-10: The `README.md` file in `diff` mode with the headshot added.

You have now made changes to your project. The only problem is these changes are still on their own branch, and not on the master branch. To find out how get your changes merged into the master branch, refer to the next section.

Merging a Pull Request

After you have all your changes in a pull request (see the preceding section), you can merge those changes into the master branch by following these steps:

1. **On the main Code tab, click the View #1 button to get to your pull request.**

   My pull request page, shown in Figure 3-11, has three different commits. The original one is when I added my headshot to my repo. The next one is when I added the new text to my README.md file, and the third is when I added my headshot to my README.md file. (If you haven't added these items and would like to, see the section “Modifying README.md”, earlier in this chapter.)

2. **Click the Files changed tab to see all the changes made to this repo.**

   Files that appear in red will be deleted, while the lines in green will be added.

3. **(Optional) To change the way you see the diff, click the Diff Settings drop-down list and then click Split then Apply and reload.**

   If you split the view, your screen changes (see Figure 3-12).

4. **In the Conversation tab, scroll to the bottom of the pull request and click the big green Merge pull request button.**

   The Confirm merge dialog box replaces the section with the Merge button from Figure 3-12, as shown in

Figure 3-13.

5. **Click Confirm merge.**

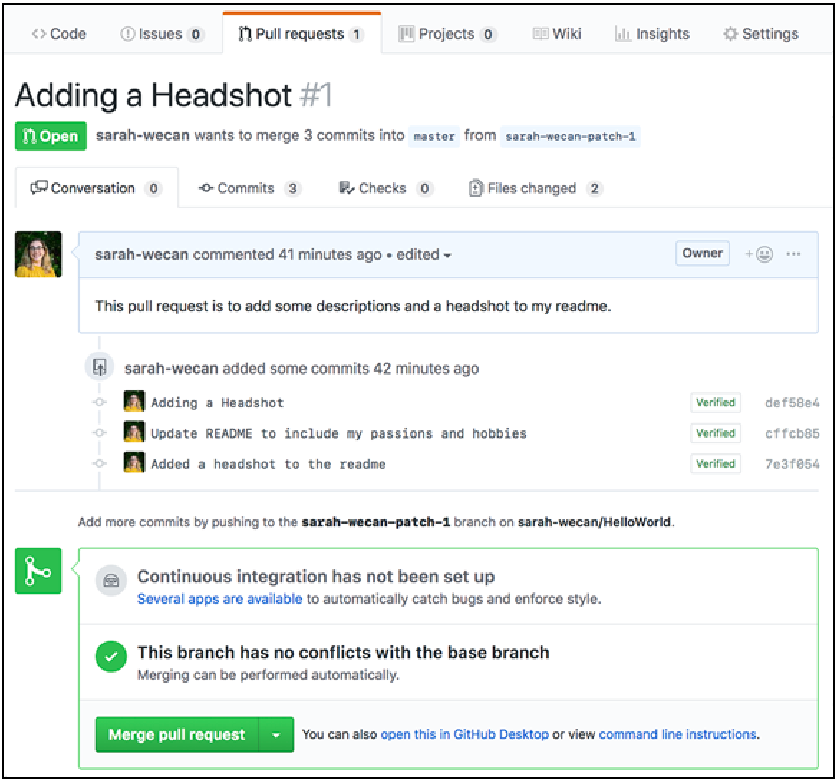
   You see a message that your pull request merge was successful, with an option to delete the branch (see Figure 3-14).

6. **Click Delete branch.**

   Your pull request is merged, and the branch is deleted. Don’t worry, if you need that branch back for some reason you can restore it. It’s nice to keep things tidy within the repository.

7. **Click the Code tab to go to your code.**

   You see the master branch with your picture and the changed README.md file.


Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
Settings

Adding a Headshot #1

Open

sarah-wecan wants to merge 3 commits into `master` from `sarah-wecan-patch-1`

Conversation 0
Commits 3
Checks 0
Files changed 2

sarah-wecan commented 41 minutes ago • edited

Owner

This pull request is to add some descriptions and a headshot to my readme.

sarah-wecan added some commits 42 minutes ago

Adding a Headshot	Verified	def58e4
Update README to include my passions and hobbies	Verified	cffcb85
Added a headshot to the readme	Verified	7e3f054

Add more commits by pushing to the **sarah-wecan-patch-1** branch on **sarah-wecan/HelloWorld**.

Continuous integration has not been set up

Several apps are available to automatically catch bugs and enforce style.

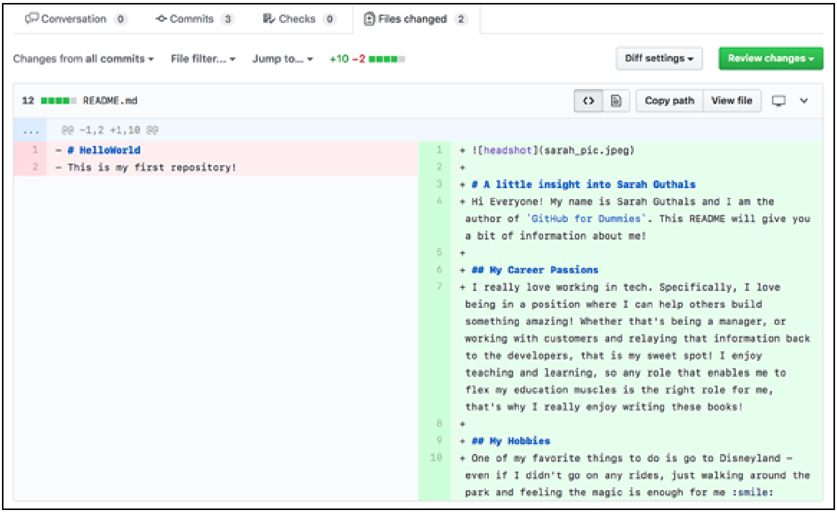
This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

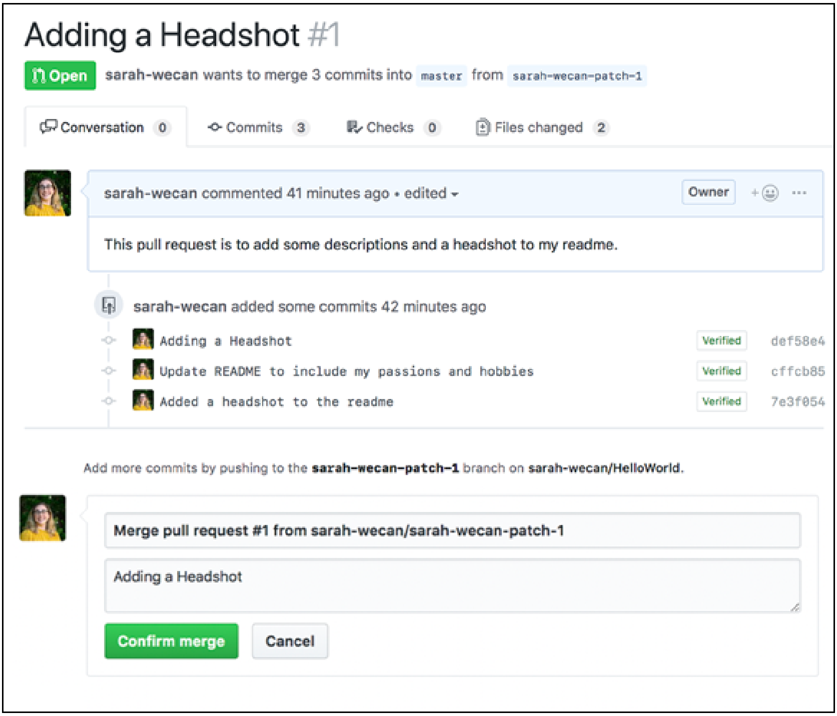
FIGURE 3-11: My pull request page.


Conversation 0
Commits 3
Checks 0
Files changed 2
Changes from all commits
File filter...
Jump to...
+10 −2
Diff settings
Review changes
12 README.md
Copy path
View file

```
@@ -1,2 +1,10 @@
- # HelloWorld
- This is my first repository!
```

```
+ ![headshot](sarah_pic.jpeg)
+
+ # A little insight into Sarah Guthals
+ Hi Everyone! My name is Sarah Guthals and I am the
 author of `GitHub for Dummies`. This README will give you
 a bit of information about me!
+
+ ## My Career Passions
+ I really love working in tech. Specifically, I love
 being in a position where I can help others build
 something amazing! Whether that's being a manager, or
 working with customers and relaying that information back
 to the developers, that is my sweet spot! I enjoy
 teaching and learning, so any role that enables me to
 flex my education muscles is the right role for me,
 that's why I really enjoy writing these books!
+
+ ## My Hobbies
+ One of my favorite things to do is go to Disneyland -
 even if I didn't go on any rides, just walking around the
 park and feeling the magic is enough for me :smile:
```

FIGURE 3-12: The diff in split view.


Adding a Headshot #1

sarah-wecan wants to merge 3 commits into `master` from `sarah-wecan-patch-1`

sarah-wecan commented 41 minutes ago • edited

This pull request is to add some descriptions and a headshot to my readme.

sarah-wecan added some commits 42 minutes ago

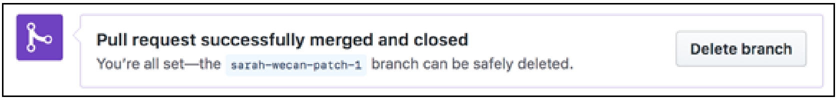
Adding a Headshot	Verified	def58e4
Update README to include my passions and hobbies	Verified	cffcb85
Added a headshot to the readme	Verified	7e3f054

Add more commits by pushing to the **sarah-wecan-patch-1** branch on **sarah-wecan/HelloWorld**.

Merge pull request #1 from sarah-wecan/sarah-wecan-patch-1

Adding a Headshot

FIGURE 3-13: The dialog box to confirm the merge of the pull request.


Pull request successfully merged and closed

You're all set—the `sarah-wecan-patch-1` branch can be safely deleted.

FIGURE 3-14: The merge confirmation for a pull request.

Using Issues and Project Boards

Issues on a GitHub repo are a great way to track the things you need to fix, add, or change. When you combine issues with project boards, you get insights into your project that would otherwise be hard to track. In this section, you create issues and project boards and change your `README.md`.

Creating a project board and an issue

To get started on issues and project boards, go to your repo on the Code tab and then follow these steps:

1. **Click the Projects tab and then click the Create a project button.**

   The Create a new project dialog box appears, as shown in Figure 3-15.

2. **Add a project name and description.**

3. **From the preloaded Template drop-down list, select a project template and click the Create project button.**

   In this example, I chose Automated kanban as my project template.

   A project board appears with some To do cards to show you how things work (see Figure 3-16).

4. **Click the Manage button at the bottom of the To do column to open the dialog box to choose what automation should happen.**

   Figure 3-17 shows the automation settings for the To do column. Every time an issue gets created, it is automatically added to your To do column. You create an issue in Step 8 of this section.

5. **Click the Manage button at the bottom of the In progress column.**

   A similar automation settings dialog box appears (see Figure 3-17). Every time an issue is re-opened after you close it, or a pull request is created or reopened, it appears in the In progress column. You see this card in the To do column if you revisit the project after Step 11 of this section.

6. **Click the Manager button at the bottom of the Done column.**

   Similar to Figure 3-17, automation settings appear. Every time an issue or pull request is closed or a pull request is merged, the cards move from the previous columns to this column. You see this in Step 14 of the section “Closing an issue”.

TIP You can change the automation settings for these columns or add other columns. Check out `https://help.github.com/articles/configuring-automation-for-project-boards` for more information on project board automation.

7. **Delete the cards that were automatically added to the To do column by clicking the three dots on the top right of each column and choosing Delete note from the drop-down menu that appears.**

   The card disappears. You can then click on the three dots of the next card that was automatically added to delete it.

8. **On the Issues tab, click New issue.**

   TIP If you haven’t previously dismissed the dialog box, Figure 3-18 shows the dialog box that appears above the Issues tab. Issues and pull requests can have labels, which are useful when you’re trying to sort through things that still need to be done. If you’re going to contribute to an open source project, looking for help wanted or `good first issue` labels is a great way to get started in the community. (see Part 5 for more information on open source software).

9. **Add a title and description to the new issue and then, on the right side, assign yourself to it.**

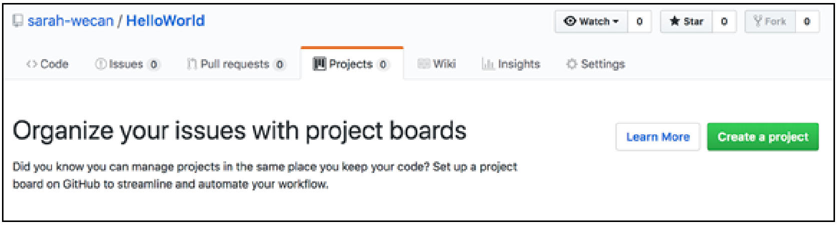
10. **Link this issue with a project by clicking Projects and choosing Tracking Changes to HelloWorld.**

    After you choose it, you see it in blue under Projects, as shown in Figure 3-19.

11. **Click the Submit new issue button.**

    Just like with the pull request, the issue has a title, number, status, description, and some metadata on the right side (see Figure 3-20). One interesting bit of information is that the number is `#2` for this issue, even though it's the first issue you've created on this repo. That is because issues and pull requests follow the same numbering, so the pull request was #1, this issue is #2, and any pull request or issue you open next will continue with #3.

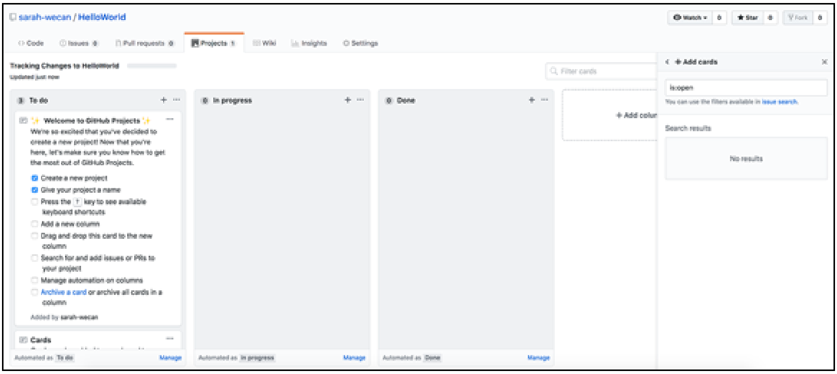
    After you create an issue, the card relating to it appears on your project board.


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Learn More
Create a project

Organize your issues with project boards

Did you know you can manage projects in the same place you keep your code? Set up a project board on GitHub to streamline and automate your workflow.

FIGURE 3-15: The page to create a new project.


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
Filter cards
To do
In progress
Done
+ Add column
Add cards
Search results
Automated as To do
Manage
Automated as In progress
Manage
Automated as Done
Manage

Tracking Changes to HelloWorld

Updated just now

✨ Welcome to GitHub Projects ✨

We're so excited that you've decided to create a new project! Now that you're here, let's make sure you know how to get the most out of GitHub Projects.

☑ Create a new project
☑ Give your project a name
☐ Press the ? key to see available keyboard shortcuts
☐ Add a new column
☐ Drag and drop this card to the new column
☐ Search for and add issues or PRs to your project
☐ Manage automation on columns
☐ Archive a card or archive all cards in a column

Added by sarah-wecan

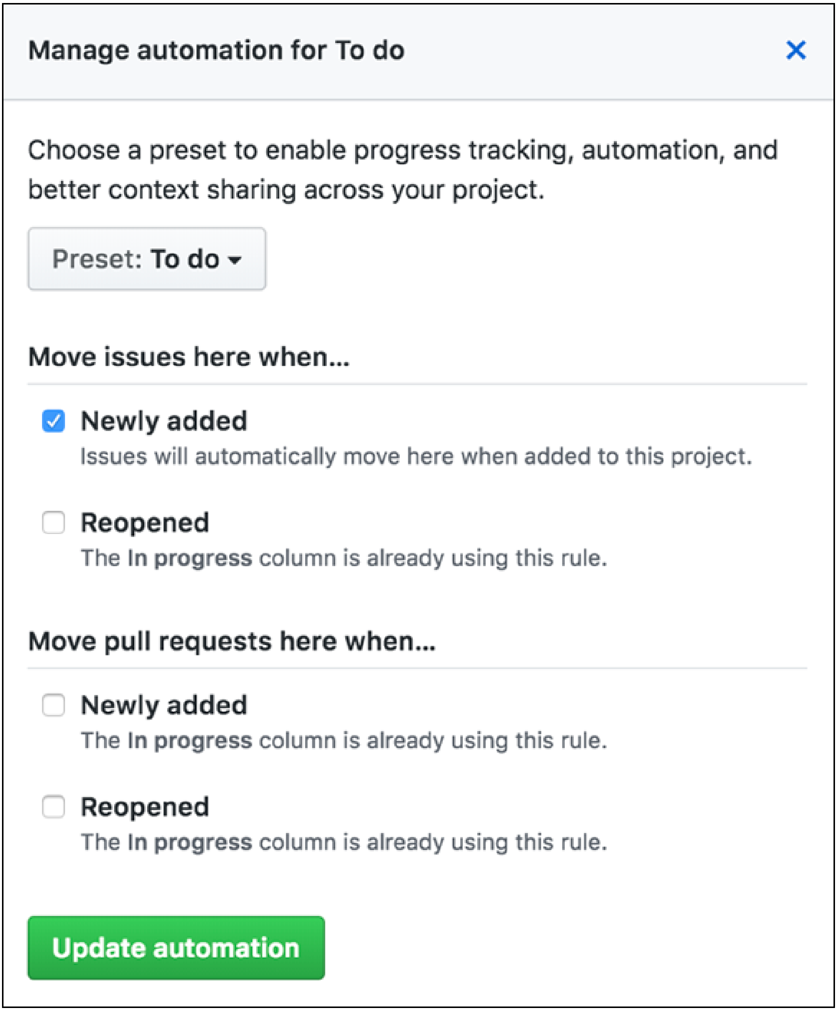
Cards

is:open

You can use the filters available in issue search.

No results

FIGURE 3-16: A blank kanban project board template.



Manage automation for To do

Choose a preset to enable progress tracking, automation, and better context sharing across your project.

Preset: **To do**

Move issues here when...

☑ **Newly added**
Issues will automatically move here when added to this project.

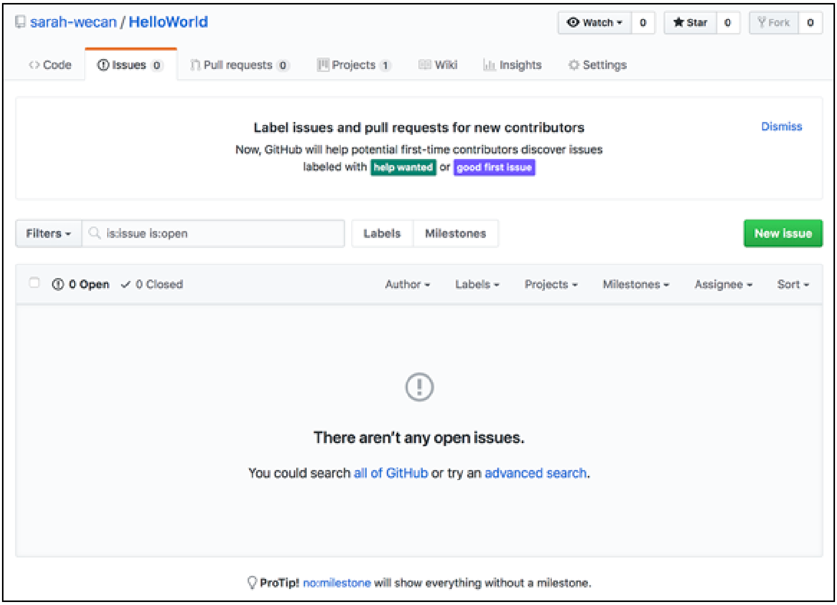
☐ **Reopened**
The **In progress** column is already using this rule.

Move pull requests here when...

☐ **Newly added**
The **In progress** column is already using this rule.

☐ **Reopened**
The **In progress** column is already using this rule.

FIGURE 3-17: The automation settings for the To do column on a kanban project.


sarah-wecan / HelloWorld
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
Dismiss
Filters
is:issue is:open
Labels
Milestones
New issue
0 Open
0 Closed
Author
Labels
Projects
Milestones
Assignee
Sort

Label issues and pull requests for new contributors

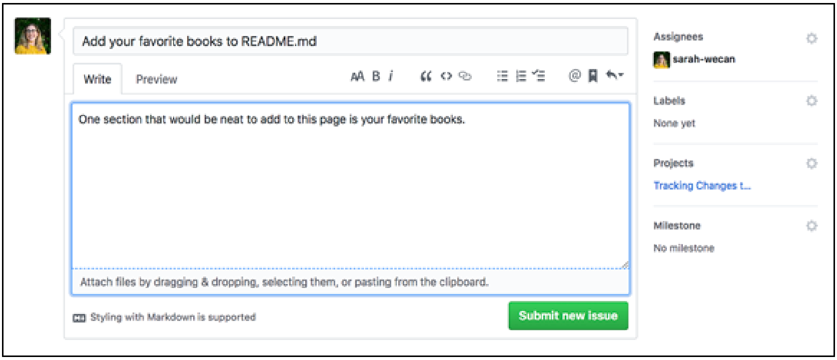
Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

There aren't any open issues.

You could search all of GitHub or try an advanced search.

ProTip! no:milestone will show everything without a milestone.

FIGURE 3-18: The page that shows all issues for the `HelloWorld` repository.


Write
Preview
Assignees
Labels
Projects
Milestone
Submit new issue

Add your favorite books to README.md

One section that would be neat to add to this page is your favorite books.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

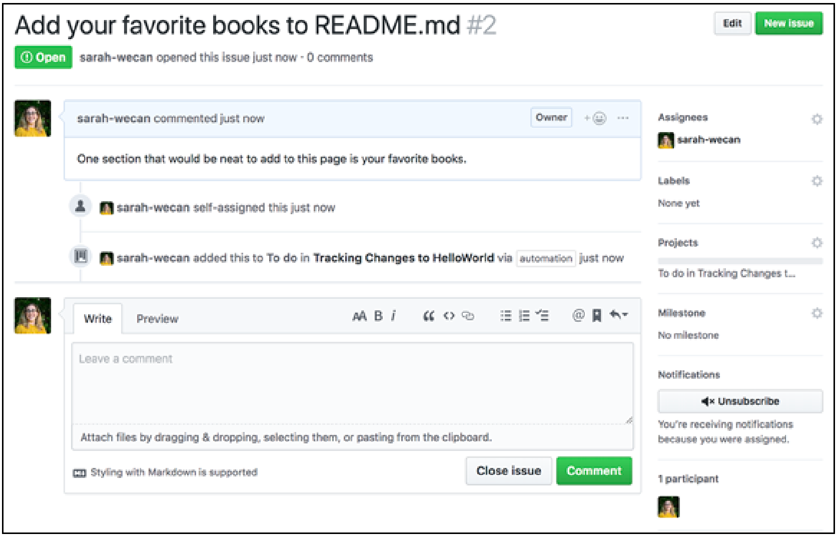
Assignees: sarah-wecan

Labels: None yet

Projects: Tracking Changes t...

Milestone: No milestone

FIGURE 3-19: A new issue linked to the project Tracking Changes to HelloWorld.



Add your favorite books to README.md #2

sarah-wecan opened this issue just now · 0 comments

sarah-wecan commented just now

One section that would be neat to add to this page is your favorite books.

sarah-wecan self-assigned this just now

sarah-wecan added this to To do in Tracking Changes to HelloWorld via automation just now

Assignees: sarah-wecan

Labels: None yet

Projects: To do in Tracking Changes t...

Milestone: No milestone

Notifications: You're receiving notifications because you were assigned.

1 participant

Edit
New issue
Open
Owner
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Close issue
Comment
Unsubscribe

FIGURE 3-20: The description page of an open issue.

To see the new card in the To do column, click on the project board link under Projects.

Closing an issue

The best way to close an issue is to create a pull request with changes to address what was written in the issue. Understanding the relationship between issues and pull requests can help you on your own projects and open source projects.

To close out the issue, follow these steps:

1. **From the Code tab on your repo, click the pencil icon for the `README.md` file to edit the file.**
2. **Add a section at the bottom of the `README.md` file with your favorite books.**
3. **Scroll to the bottom of the page and add a title to the commit.**
4. **Choose Create a new branch for this commit and start a pull request.**
5. **Click Commit changes.**
6. **Add a description to the pull request.**

   Specifically, make sure that you write `closes #2` on its own line. When you type `#`, GitHub suggests any issue or pull request that you have in this repo to auto-fill.

7. **Link the project to this pull request.**

   You link the project to the pull request the same way that you link a project to an issue. (See step 10 in the section “Creating a project and an issue”, earlier in this chapter.)

8. **Click Create pull request.**

9. **Return to your project board.**

   You see the card for the pull request in the In progress column, as shown in Figure 3-21.

10. **Click the pull request card title to preview the pull request.**

    Click Go to pull request for full details to return to the pull request.

11. **Click the Files changed tab and then click View file.**

    You see the `README.md` file with your last section added (see Figure 3-22).

12. **Click the Back button on your browser.**

13. **If you're happy with the changes, click the Conversation tab, click Merge pull request button, click Confirm merge button, and then click the Delete branch button.**

    You can revisit the section “Merging a pull request,” earlier in this chapter, for more details on how to complete this step if you get stuck.

14. **Click the Projects tab and choose the project that you created.**

Both of the cards have moved from the To do and In progress columns to the Done column.

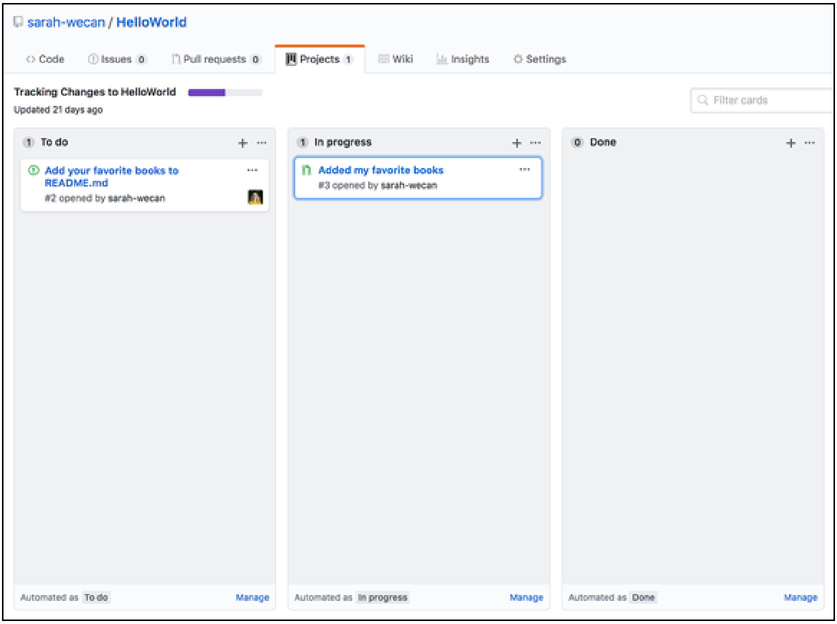
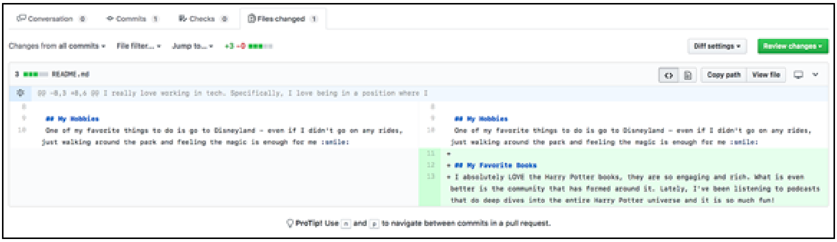

sarah-wecan / HelloWorld
Code
Issues 0
Pull requests 0
Projects 1
Wiki
Insights
Settings
Tracking Changes to HelloWorld
Updated 21 days ago
Filter cards
1 To do
Add your favorite books to README.md
#2 opened by sarah-wecan
1 In progress
Added my favorite books
#3 opened by sarah-wecan
0 Done
Automated as To do
Manage
Automated as In progress
Manage
Automated as Done
Manage

FIGURE 3-21: The project board with an open issue and an open pull request.


Conversation 0
Commits 1
Checks 0
Files changed 1
Changes from all commits
File filter...
Jump to...
+3 −0
Diff settings
Review changes
3 README.md
Copy path
View file
ProTip! Use n and p to navigate between commits in a pull request.

```
@@ -8,3 +8,6 @@ I really love working in tech. Specifically, I love being in a position where I

## My Hobbies
One of my favorite things to do is go to Disneyland - even if I didn't go on any rides,
just walking around the park and feeling the magic is enough for me :smile:
+
+## My Favorite Books
+I absolutely LOVE the Harry Potter books, they are so engaging and rich. What is even
better is the community that has formed around it. Lately, I've been listening to podcasts
that do deep dives into the entire Harry Potter universe and it is so much fun!
```

FIGURE 3-22: The diff of `README.md` showing the added text.

Chapter 4

Setting Up a GitHub Website Repo

IN THIS CHAPTER

- **Getting acquainted with GitHub Pages**
- **Creating a project website**
- **Preparing a website repo**
- **Preparing your local computer**
- **Discovering resources for building your website on GitHub.com**

One technology that truly pushed the boundaries of society and software development was the Internet. As the Internet became more embedded in everyday life, it brought new meaning, career opportunities, and ways to connect for millions of people. A more recent phenomenon enabled by the Internet is a set of sites categorized as social media. Social media profiles provide a way for people to express who they are, what their interests are, and how to connect with them. But social media creates new challenges.

Personally, I have accounts on Twitter, Instagram, Snapchat, Facebook, and LinkedIn. I don’t feel comfortable sharing personal accomplishments (like having a baby) on my LinkedIn, but also rarely share minor professional accomplishments (like starting a new project) on my Instagram. For me, having a website gives me a place to point people to all of me, not just the version of me that fits the community of the particular platform I’m using. That way, if folks want to learn more about the new project I’m working on, they will head to my website where they may also stumble upon the fact that my daughter is now walking and reach out to me to suggest a walker that worked well for their child. This interaction improves my bonds and connections with people from all aspects of my life.

Others may use a custom website to still focus on a particular part of who they are, but they like having the control over what they share and how they do it. This chapter, along with Chapter 5, guides you through turning any project repo that you own into a website and creating your own website on GitHub.com. In a matter of minutes, you can have a website up and running without having to pay for additional services.

Introducing GitHub Pages

GitHub Pages is a fast and easy way to make a website that is hosted on GitHub.com. The code in your repo will be the code running the website. Even better is that it's much easier to style your websites with Jekyll, a free, open-source site generator that takes Markdown files and creates websites with support for themes.

TIP You can discover more about Jekyll at `https://jekyllrb.com` or even check out what it's up to on its GitHub repo (`https://github.com/jekyll/jekyll`).

With GitHub Pages, you can create a website using Markdown or HTML/JavaScript/CSS.

REMEMBER If you need help remembering what you can do with Markdown, visit the Markdown GitHub Guide at `https://guides.github.com/features/mastering-markdown`.

Turning a Project Repo into a Website

GitHub Pages is a great tool that is integrated into GitHub.com. GitHub Pages will look for a README.md file on your master branch and use it as the landing page for your website, meaning you don't have to do much to get it up and running! Just follow a couple short steps:

1. **Open a repository on GitHub.com.**

   For this example, I use my `HelloWorld` repo that I create in Chapter 3.

2. **On the home page for the repo, click the Settings tab on the top right to open the Settings page.**

   The Options Settings page, shown in Figure 4-1, appears.

3. **From the Select source drop-down menu, change the source for GitHub Pages from `none` to `master branch` and then click Save.**

   The web page refreshes. If you scroll back to the GitHub Pages section, you see `master branch` under Source.

4. **Choose a theme for your website.**

   A new web page loads, as shown in Figure 4-2. Browse through the themes and choose one. For my example website for this book, I use the Time Machine theme.

5. **After you choose a theme for your website, click Select theme.**

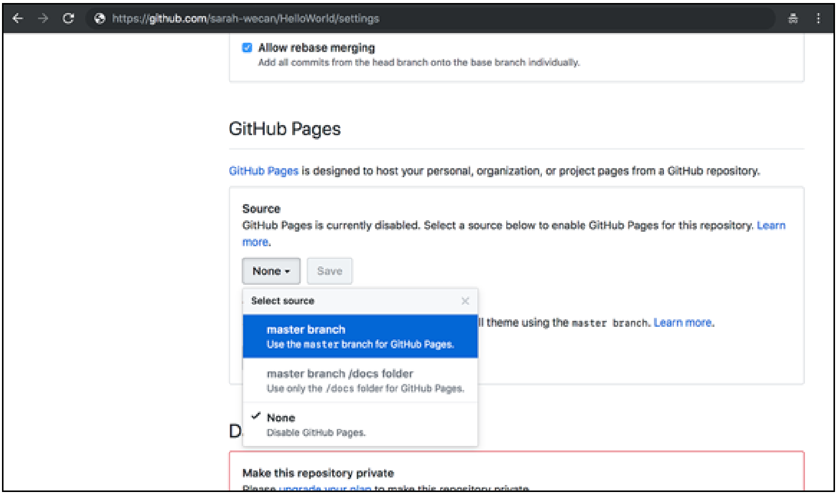
   You return to the Settings page.

6. **Visit your project website.**

   Scroll back down to the GitHub Pages section on the Settings page for your repository, and you see a notification in green that your site has been published with a clickable URL.

   TIP If you don't see that your site has been published, try refreshing the web page. Sometimes browsers will cache, and these changes won't appear immediately.

7. **Click the URL and view the website you just made.**



https://github.com/sarah-wecan/HelloWorld/settings

☑ Allow rebase merging
Add all commits from the head branch onto the base branch individually.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more.

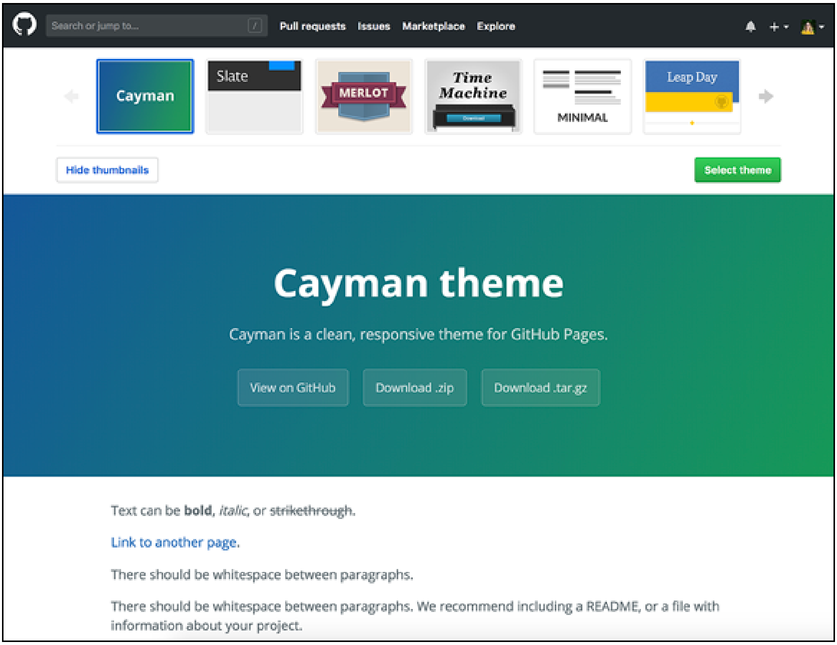
None
Save
Select source
master branch
Use the master branch for GitHub Pages.
master branch /docs folder
Use only the /docs folder for GitHub Pages.
✓ None
Disable GitHub Pages.

ll theme using the `master branch`. Learn more.

D

Make this repository private

FIGURE 4-1: The GitHub Pages section under repository settings.



Search or jump to...
Pull requests
Issues
Marketplace
Explore
Cayman
Slate
MERLOT
Time Machine
MINIMAL
Leap Day
Hide thumbnails
Select theme
View on GitHub
Download .zip
Download .tar.gz

Cayman theme

Cayman is a clean, responsive theme for GitHub Pages.

Text can be **bold**, *italic*, or ~~strikethrough~~.

Link to another page.

There should be whitespace between paragraphs.

There should be whitespace between paragraphs. We recommend including a README, or a file with information about your project.

FIGURE 4-2: The GitHub Pages website theme chooser.

Setting Up a Personal Website Repo

To create a new repo that houses your own personal website, you need to set up your repo:

1. **Create a new repository and name it *`username`*`.github.io`, where *username* is replaced with your actual GitHub username.**

   For example, the name of Sarah's repository is `sarah-wecan.github.io`. If you're unsure how to create a new repository, see Chapter 3.

2. **Make the repository public, initialize it with a README, choose a license, if you want, and then click Create Repository.**

   If you're unsure whether you want a license, see Chapter 3, which describes the benefits of choosing a license.

   The page refreshes to the home page of your new repository.

3. **Create a project with the Automated kanban template.**

   If you don't know how to create a project, see Chapter 3 for guidance.

4. **At the top of your repo again, click the Settings tab and scroll down until you see the GitHub Pages section.**

   Notice that it says your site is already ready to be published at a certain URL (mine is `https://sarah-wecan.github.io/`). If you go to this URL, you should see a simple barebones web page with the contents of your README.md file. If you get a 404, just wait a moment and refresh the page. It can take a few seconds for GitHub Pages to build your site.

5. **In the GitHub Pages section, click Choose a theme and then click Select theme.**

   If you’ve already chosen a theme and want to change it, this button is labeled Change theme. After you choose a theme, you’re taken to one of two pages, depending on which theme you chose. Either you’re taken back to the Settings page, from which you can navigate back to your code and you see a `_config.yml` file with your theme information. In this case, you can skip Step 6. Alternatively, you're taken to the README.md file in editing mode, where the theme information has been added to the README. In this case, you should follow Step 6. In either case, the commit of the `_config.yml` file happens automatically, and that commit triggers GitHub Pages to rebuild your site with the theme applied.

6. **Commit your README.md file.**

   Your README.md file has now been modified to include information about your Jekyll template. Scroll to the bottom, add a title to the commit message, and choose Create a new branch for this commit and start a pull request. Then click Propose file change. The page refreshes with a new pull request ready to be created. You can merge your pull request. Then you can navigate back to your code, and you see the `_config.yml` file as described in Step 5.

7. **Create a pull request with a more inclusive title.**

   For instructions on creating a pull request, see Chapter 3.

Change the title of the pull request to describe everything you want to do for this iteration of your website and add a description. For example, I added the following list:

```
To create a basic website:
- [ ] Add a theme
- [ ] Add an index.md file with "Hello World" on it
```

8. **Link this pull request to the project and create the pull request.**

   To ensure your project board automatically tracks the progress of your website, choose the project created in Step 3 on the right side of the pull request from the Projects section. Then click Create pull request.

9. **Update the pull request when appropriate.**

   TIP The description of a pull request is not static. When you first create a pull request, you may have a list of things you want to do before merging the code into the master branch. You may also end up making changes throughout. Make sure to always revisit the description and make sure it is accurate. For example, if you're following these steps exactly, you have already chosen a theme, so you can check the first box.

10. **Verify the project board automation.**

    Go back to the project board by clicking on the Projects tab at the top of your repo and notice that a new card is in the In progress column. This card has a checklist on it, just like the description of the pull request does, and you can see that one of two tasks has already been completed.

11. **Switch to the pull request branch.**

    Go back to the Code tab of your repo and switch to the branch that is associated with your pull request. Mine is `sarah-wecan-patch-1`.

    TIP If you don't know how to switch to a different branch on GitHub.com, see Chapter 3.

12. **Create an `index.md` file.**

    On the top right of your code file list, click Create new file. Name the file `index.md` and add a header that says "Hello World!":

    ```
    # Hello World!
    ```

    Commit that file to the same branch you've been working on, with a title and commit message.

13. **Update and merge the pull request.**

    Go back to the pull request and check off the second check box. Because everything that you wanted to do for this pull request has been completed, the pull request can be merged. Click Merge pull request, Confirm merge, and then Delete branch.

14. **Verify the project board automation.**

    The pull request shows in the conversation that it has been moved from the In progress column to the Done column. You can also go back to the project board. Notice that the card has moved to the Done column, and both checklist items have been completed.

15. **Verify the website was published.**

Go back to your URL (mine was `https://sarah-wecan.github.io`). You have a working website with the theme you chose.

TIP If you don't see the theme that you chose, try refreshing your web page. You now have a website that you can continue building and customizing as you do more and have more to share with the world.

Creating Issues for Your Website

After you have a GitHub.com website repository (see preceding section), you can think through the sections you want to have on your website. Creating issues for everything you want to add or change about your website can help you plan and remember all the little things you want to change.

Say that someone gives you a great suggestion. You don't want to pull out your computer and make the change right then and there, but you can quickly jump on GitHub.com and create an issue to remind you to add it later. Creating an issue can also be useful if you are working on your website, and you've found something that is bothering you that you want to change. Instead of derailing what you're already working on with a new task, you can just make a quick issue and get to it later.

To get started with this planning phase, go to the Issues tab on your repo and click New issue. Create an issue for all the things you want to add to your website.

In this book, I create four issues to use in my example:

- **Change the title and tagline.** The title and tagline of your website is currently something auto-generated. You probably want to change it to your name and some tagline that represents you. Create an issue, assigning yourself to it and linking it to your project board:

```
Issue Title: Change the title and tagline
Issue Description: Make the title and tagline
   something unique to you.
```

- **Add sections to the website.** Without even having to leave an issue you created, you can click New issue and create another issue to add in sections to your website, assign yourself to the issue, and link it to your project board. In my example, I've chosen to add three sections:

  ```
  Issue Title: Add a couple of sections to the website

  Issue Description: Add three sections to the website:

  - [ ] About Me

  - [ ] Contact Information

  - [ ] Blog
  ```

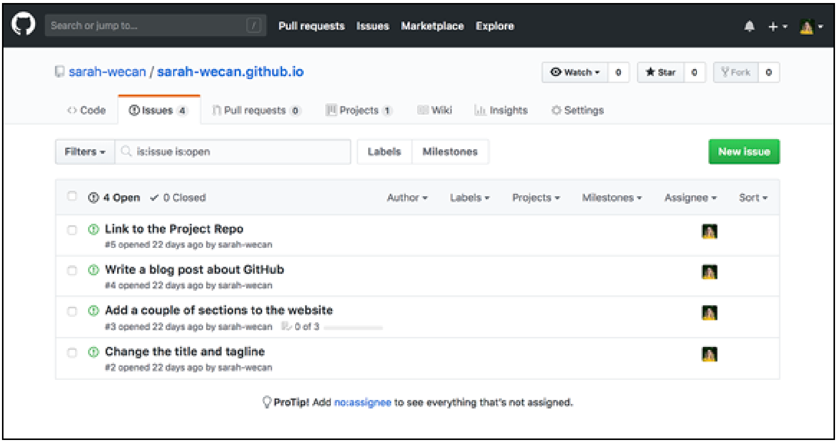
- **Write a blog post.** Because one of the sections is to add a blog to the website, you may also want to actually publish your first blog post. Make a new issue to remind yourself, assign yourself to the issue, and link it to your project board:

  ```
  Issue Title: Write a blog post about GitHub

  Issue Description: With a blog section ready to go, add a blog post that talks about your interactions with GitHub so far.
  ```

- **Link your project repo.** You can turn your project repo into a website, as described earlier in this chapter. Why not link that website from your personal website? Create an issue, assign yourself to the issue, and link it to your project board. Later, you can link to other projects that you contribute to or find interesting:

  ```
  Issue Title: Link to the Project Repo

  Issue Description: I've turned http://github.com/sarah-wecan/HelloWorld into a website and I want to link that website from this one.
  ```

Four issues now appear under the Issues tab of my website repo, as shown in Figure 4-3. They also appear on my project board in the To do column (see Figure 4-4).


Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / sarah-wecan.github.io
Watch 0
Star 0
Fork 0
Code
Issues 4
Pull requests 0
Projects 1
Wiki
Insights
Settings
Filters
is:issue is:open
Labels
Milestones
New issue
4 Open
✓ 0 Closed
Author
Labels
Projects
Milestones
Assignee
Sort

Link to the Project Repo
#5 opened 22 days ago by sarah-wecan

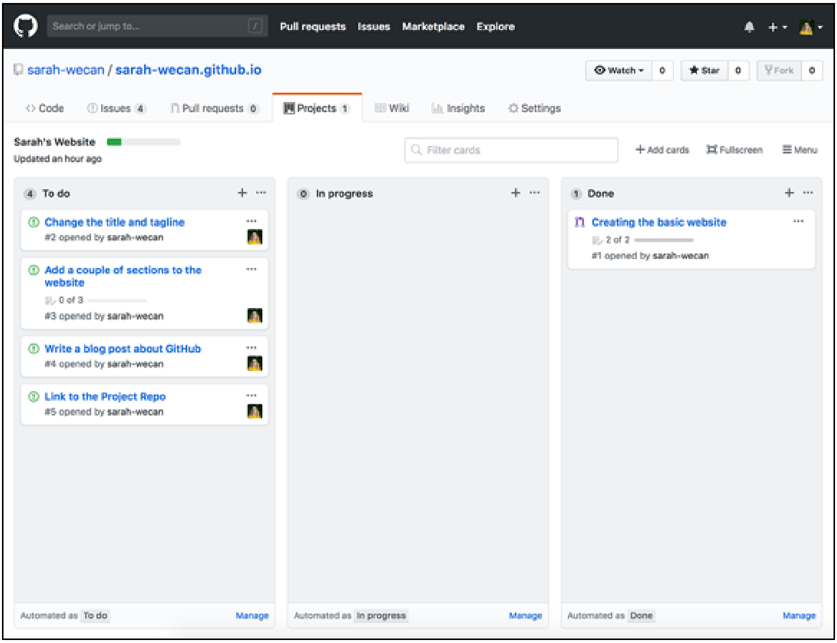
Write a blog post about GitHub
#4 opened 22 days ago by sarah-wecan

Add a couple of sections to the website
#3 opened 22 days ago by sarah-wecan 0 of 3

Change the title and tagline
#2 opened 22 days ago by sarah-wecan

ProTip! Add no:assignee to see everything that's not assigned.

FIGURE 4-3: The issue list for the website repository.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / sarah-wecan.github.io
Watch 0
Star 0
Fork 0
Code
Issues 4
Pull requests 0
Projects 1
Wiki
Insights
Settings
Sarah's Website
Updated an hour ago
Filter cards
+ Add cards
Fullscreen
Menu
4 To do
0 In progress
1 Done
Automated as To do
Manage
Automated as In progress
Manage
Automated as Done
Manage

Change the title and tagline
#2 opened by sarah-wecan

Add a couple of sections to the website
0 of 3
#3 opened by sarah-wecan

Write a blog post about GitHub
#4 opened by sarah-wecan

Link to the Project Repo
#5 opened by sarah-wecan

Creating the basic website
2 of 2
#1 opened by sarah-wecan

FIGURE 4-4: The project board for the website repository.

Setting Up Your Local Environment

This section assumes you already set up GitHub Desktop and Atom. If you haven't, Chapter 2 can help guide you through this process.

In this section, you get your website working so that you can modify files on your local computer instead of on GitHub.com.

TIP Modifying files on your local computer can be useful if you need to work on a project when you won't have Internet access or if you need to browse a lot of files while editing files.

Cloning a repo in GitHub Desktop

The first step in modifying files on your local computer is to get your website repo onto your computer:

1. **Open GitHub Desktop and choose File ⇒ Clone a Repository on the menu bar.**

   You see a dialog box with three tabs; GitHub.com, Enterprise, and URL.

   TIP A nice alternative approach to cloning a repository when you have GitHub Desktop is to click the Clone or Download button on the home page of every repository. When you click the button, you see a flyout that includes an Open in Desktop button. Click that button to launch GitHub Desktop (if it's not already running) and clone the repository to your local machine.

2. **On the GitHub.com tab of the Clone dialog box, your repositories auto-fill for you from GitHub.com.**

WARNING If your GitHub.com repositories don't auto-fill in the Clone dialog box, it probably means you're no longer signed in to GitHub.com in GitHub Desktop. You can log in by choosing GitHub Desktop ⇒ Preferences from the top menu bar. Click the Accounts tab and sign in. If it appears you're logged in but your repos are still not showing up, try signing out and signing back in.

3. **Choose your personal website repository and choose where you want it to be stored on your local machine.**

   I chose the default path as the place to store the repository on my local machine.

4. **Click Clone.**

   GitHub Desktop refreshes with your repo information included.

Touring GitHub Desktop

GitHub Desktop offers a variety of features to help you with your development and interactions with GitHub.com. You can check out the GitHub Desktop User Guides at `https://help.github.com/desktop` if you need additional support beyond this book. Figure 4-5 highlights the top six features:

- **Repository list:** As you clone more repositories to your local computer, clicking the Current Repository drop-down list reveals all the repositories that you have on your local computer, enables you to quickly switch between them, and gives you a quick button to add a new one.
- **Branch list:** The branch list gives you a quick overview of all the branches that you have checked out on your local computer, as well as a button to quickly create a new branch.
- **Pull request list:** One the same drop-down list as the branch list, you see a second tab that lists all the pull requests that are open on

this repo.

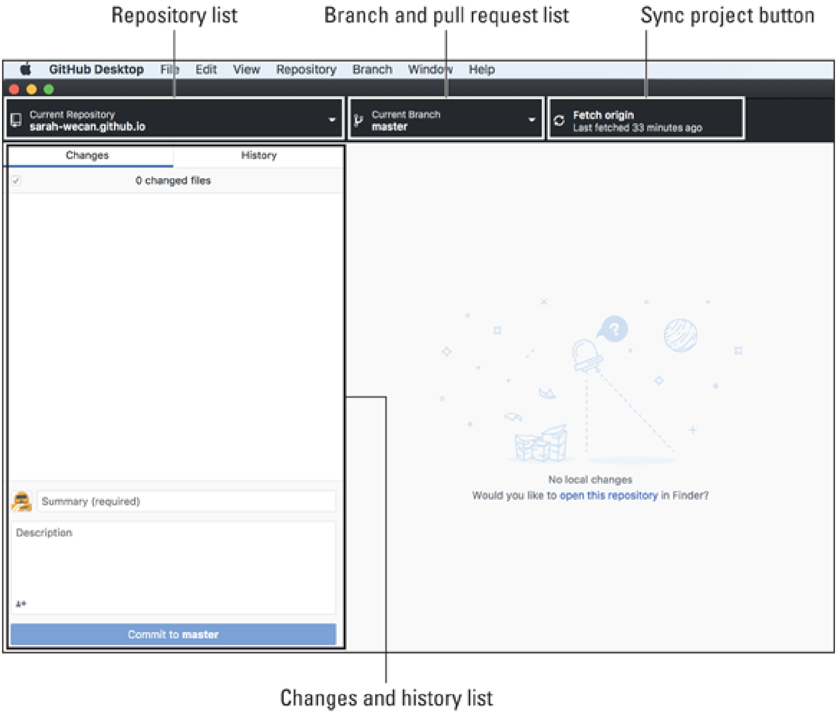
- **Sync project button:** As you start to make changes and/or changes are made on the repo outside of what you’re doing on your local machine, you will need to sync. Because Desktop hasn’t detected any changes made on GitHub.com or your local computer, the option presents itself as a `fetch` to start. If you start to make changes on your local machine, you can choose to push your local changes to GitHub.com. If you start to make changes on GitHub.com, you can choose to `pull` those changes to your local machine. If you create a repository on your local machine and it isn't on GitHub.com yet, you can choose to `publish` your project to GitHub.com.

  WARNING If you do not push your changes to GitHub.com, they won’t be available for other people and if your computer were to crash, you would lose all your work. We highly recommended that you `push` your code often.

- **Changes list:** As you start to make changes to your code, the files that you've added, deleted, or modified show up in this changes list. You can click each file to see the diff to the right. When you’re ready to commit to those changes, you can add a Summary and Description and click the Commit to master button. At that point, you can push your changes to the branch that you’re on using the Sync project button.

  WARNING You should always double check which branch you’re on before you commit and push your changes. You can undo commits and pushes, but avoiding it is best because the process can get hairy really quickly.

- **History list:** Next to the changes you find the history of this repo. The history includes activity from your local machine that has been synced with GitHub.com and activity from GitHub.com that you may have never done on your local machine. When you click one of the events, you see a list of activity that happened.


Repository list
Branch and pull request list
Sync project button
GitHub Desktop
File
Edit
View
Repository
Branch
Window
Help
Fetch origin
Last fetched 33 minutes ago
Changes
History
0 changed files
Summary (required)
Description
Commit to master
Changes and history list

Current Repository
sarah-wecan.github.io

Current Branch
master

No local changes
Would you like to open this repository in Finder?

FIGURE 4-5: An overview of GitHub Desktop.

Opening your repo in Atom

To edit your files on your local computer, you can use a number of applications, including Atom, Visual Studio Code, or even TextEdit. In this book, we use Atom.

To open your repo in Atom:

1. **Open Atom.**

You see a blank window.

2. **Choose File ⇒ Add Project Folder from the top menu bar.**

   A file finder dialog box appears.

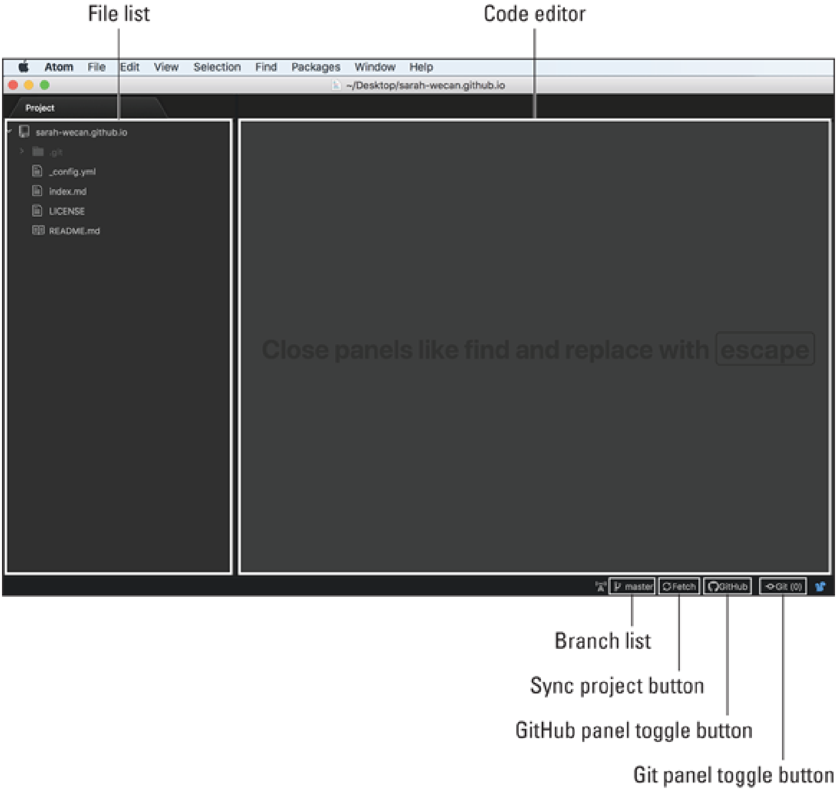
3. **From the file chooser, open the folder for your repo and click Open.**

   Your project is now open in Atom.

Touring Atom

Atom is primarily a code editor, but it also has features that make coding on GitHub repos much easier. If this section doesn't offer enough detail for everything you can do with Atom, make sure that you check out the docs. In particular, the Atom Flight Manual is always updated to reflect the newest features (`https://atom.io/docs`). Figure 4-6 shows the top six features:

- **File list:** On the left side of Atom is a list of all the files that you have in your repo. If you click a file, it opens in the center code editing area.
- **Code editor:** To the right of the file list is the code editor where you can write or modify code.
- **Branch list:** At the bottom right corner of Atom is a branch chooser. Right now, you are on the master branch of your repo. If you click the branch, a menu pops up, allowing you to choose between branches or create a new one.
- **Sync project button:** Atom supports syncing your project with what is on GitHub.com.
- **GitHub panel toggle button:** If you click the GitHub button, the GitHub panel appears in your Atom window. This panel specifically lists and supports pull request actions.
- **Git panel toggle button:** If you click the Git button, the Git panel appears in your Atom window as a second tab next to the GitHub panel. This panel specifically supports staging and committing changes, as well as viewing the history of changes on this repo.


File list
Code editor
Atom File Edit View Selection Find Packages Window Help
~/Desktop/sarah-wecan.github.io
Project
sarah-wecan.github.io
.git
_config.yml
index.md
LICENSE
README.md
master
Fetch
GitHub
Git (0)
Branch list
Sync project button
GitHub panel toggle button
Git panel toggle button

Close panels like find and replace with escape

FIGURE 4-6: An overview of Atom.

Finding Resources for GitHub Pages

Some amazing folks over at GitHub have dedicated all their time to supporting GitHub users in discovering and learning about all the features that GitHub offers. Beyond the static documentation, the GitHub Training Team offers Learning Labs.

Learning Labs are a self-guided, automated tutorial that actually has you doing the actions on GitHub.com and not just watching a video or reading a tutorial. You can find great ones for GitHub Pages, all GitHub fundamentals, Markdown, HTML, and even running your own Open Source Community. Head over to `https://lab.github.com`, authenticate with your GitHub credentials, to try some of these outlearning Labs.

Chapter 5

Creating a Website with GitHub Pages

IN THIS CHAPTER

- **Reorienting yourself with an existing project**
- **Getting ready to contribute to an existing project**
- **Adding headers and sections to a website**
- **Creating a blog using GitHub Pages**

In this chapter, you can find effective strategies for reorienting yourself with an existing project, as well as specifics on building a website with GitHub Pages. The examples shown in this chapter assume that you have an existing GitHub Pages repository. If you don't and want guidance in setting one up, see Chapter 4.

You can follow along in this chapter in order to build a simple website on GitHub Pages or jump around to the various sections to a find out how to do a specific task.

Jumping into an Existing GitHub Project

Whether you're revisiting a project that you started yesterday, one you worked on last year, or finding a new project that you've never worked on, there are quick and easy ways to get oriented with a GitHub project. In this section, you see examples of reorienting with the GitHub Pages website repo that we create in Chapter 3, but the tips apply to any GitHub repo.

To get started, make sure that you've opened your browser to

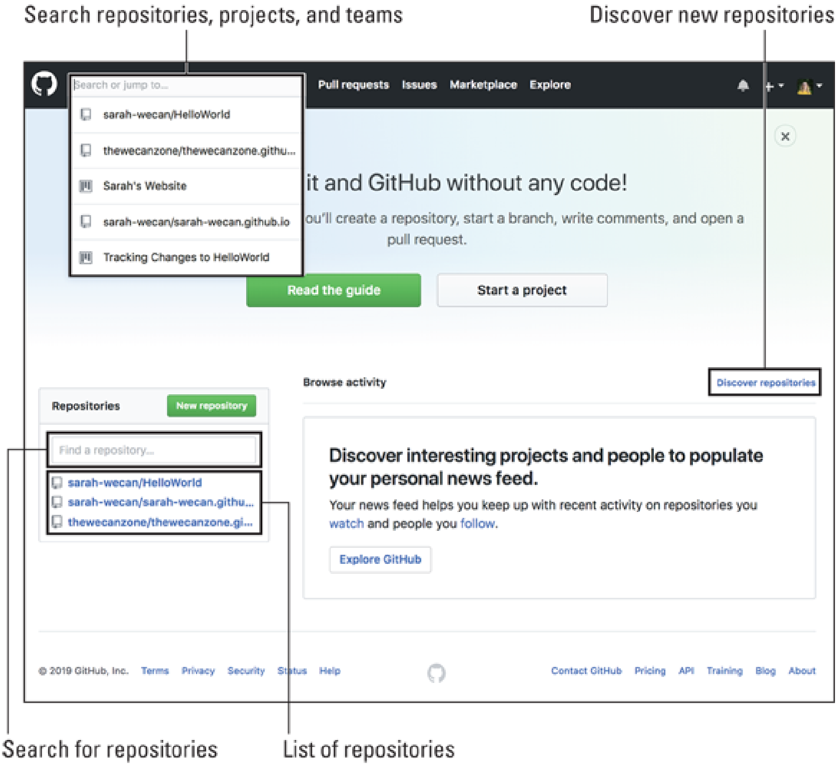
GitHub.com and have signed in. If you need to create a GitHub.com account, you can read about how to do so in Chapter 1.

Accessing the GitHub.com repo

On the left side of the GitHub.com home page is the list of repositories that you have recently opened, contributed to, or created. Directly above the list is a search bar for searching repositories. This search bar becomes more useful as you interact with more GitHub repositories because the list of repositories can grow large, especially if you belong to a big organization.

At the top of the home page is another search bar that you can use to search for repositories, project boards, and teams (a feature of organizations beyond the scope of this book). By default, this top search bar is scoped to your current context. If you're on GitHub's home page, it searches all of GitHub. If you navigate to a repository on GitHub.com, the top search bar searches within that repository.

The search bar always gives you the option to search all of GitHub.com no matter where you are. Figure 5-1 shows the three ways to find a specific repository you may be looking for and two places to find new repositories.


Search repositories, projects, and teams
Discover new repositories
Search or jump to...
sarah-wecan/HelloWorld
thewecanzone/thewecanzone.githu...
Sarah's Website
sarah-wecan/sarah-wecan.github.io
Tracking Changes to HelloWorld
Pull requests
Issues
Marketplace
Explore
Read the guide
Start a project
Browse activity
Discover repositories
Repositories
New repository
Find a repository...
sarah-wecan/HelloWorld
sarah-wecan/sarah-wecan.githu...
thewecanzone/thewecanzone.gi...
Explore GitHub
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About
Search for repositories
List of repositories

it and GitHub without any code!

ou'll create a repository, start a branch, write comments, and open a pull request.

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch and people you follow.

© 2019 GitHub, Inc.

FIGURE 5-1: Places to find GitHub repos on the GitHub.com home page.

After you find the repository you want to start collaborating with, click it. The repository's home page appears.

TIP

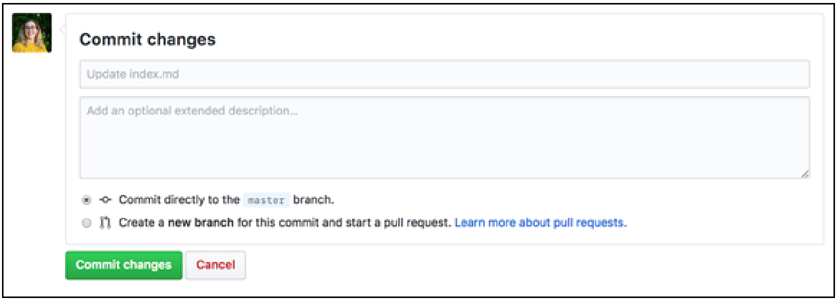
This chapter gives specific examples about contributing to the GitHub Pages website repo that you're the owner of, so if you want to follow on specifically for that, choose the repo titled *`your-username`*`.github.io`. The repo we use in this example is `sarah-wecan.github.io`.

Verifying your permissions for the repo

If you're the owner or admin of a repository, you will see a Settings tab at the top of the repository's home page. You will have complete control over the repository, including the ability to

- Invite collaborators
- Change the visibility of a repository from public to private or from private to public
- Delete the repository
- Archive the repository

If you don't see the Settings tab, you're not the owner/admin, but you may have write permissions. To determine whether you do, you can attempt to make a change to a file by navigating to the file and clicking the pencil icon. If you're able to make a change and are presented with a Commit changes box similar to the one shown in Figure 5-2, you have write permissions and were added as a collaborator for the project.



Commit changes

Update index.md

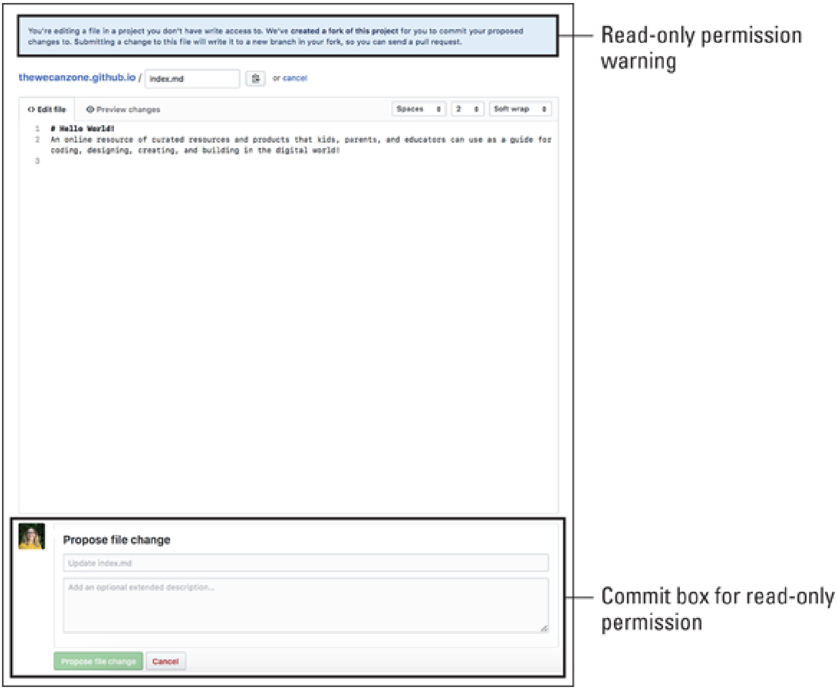
Add an optional extended description...

● Commit directly to the `master` branch.

○ Create a new branch for this commit and start a pull request. Learn more about pull requests.

FIGURE 5-2: Commit changes box when you have write permissions on a repo.

If you attempt to make a change but see the warning and commit box shown in Figure 5-3, you do not have write permissions and are not a collaborator. You can still create issues and propose file changes to the repo, but you have to get approval from someone with more permissions than you. If you're interested in how to contribute to projects that you don't have write access to, see Chapter 6.



You're editing a file in a project you don't have write access to. We've **created a fork of this project** for you to commit your proposed changes to. Submitting a change to this file will write it to a new branch in your fork, so you can send a pull request.

```
# Hello World!
An online resource of curated resources and products that kids, parents, and educators can use as a guide for coding, designing, creating, and building in the digital world!
```

Read-only permission warning
thewecanzone.github.io /
index.md
or cancel
Edit file
Preview changes
Spaces
2
Soft wrap
Propose file change
Update index.md
Add an optional extended description...
Propose file change
Cancel
Commit box for read-only permission

FIGURE 5-3: Warning and commit box when you have read-only permissions on a repo.

Orienting yourself with the project

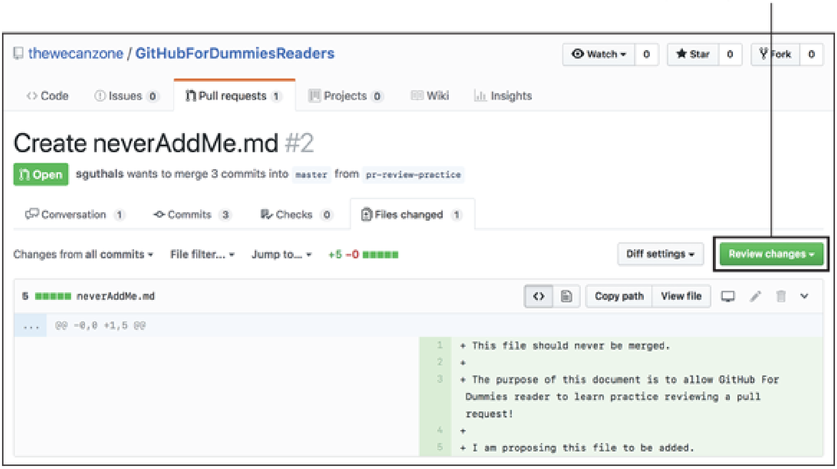
Whether you're the owner, a contributor, or a new visitor to a repo, you need to orient yourself with the project before you start working in case additional issues, pull requests, or updates have been made since your last visit. Then you're the only author on a private repo, review what needs to be done before you start so that you can begin something of the appropriate scope given the time you have to work on the project. You don't want to start building an entire new feature set if you only have a couple of hours; exploring a bug or making some minor updates may be better options.

To get oriented, or re-oriented, with a project, you should review four places on the repo:

- **Read the `README.md`, `CONTRIBUTING.md`, and `CODE_OF_CONDUCT.md` files.** Unless you're working on your own project, you should always read through these three files at least to make sure that you understand how to set up the project on your computer, contribute effectively to the project, and interact with other contributors. Not every project will have all these files, but if they exist, you should read through them to ensure that you’re a positive member of the project’s community. You can find a good example of these documents on the GitHub Desktop open source project at `https://github.com/desktop/desktop`.

- **Survey project boards.** If the project uses project boards, look at those boards. Click the Projects tab at the top of the repo and choose a project board. Each project board usually has a column of things in progress or that need to be done. If the project board has automation, then any changes to issues or pull requests (including new ones) will appear on the project board. In the example for this book, the Sarah’s Website project board has four issues listed in the To do column.
- **Read through issues.** Especially on active projects, folks are likely to have opened new issues. Click the Issues tab at the top of the repo to see the list of open issues. You can sort the issues by most recently updated to see whether folks have commented on existing issues. The default sorting is by newest (meaning the issues that were most recently opened). If you’re the repository owner, triage any new or updated issues. *Triaging* is when you sort and order items. (See the sidebar called “Triaging issues” for more information about how and why to triage.) The example for this book shows four issues that we opened to help track the progress on our website. No one else has opened up any new issues or commented on them since we last visited the repo.
- **Review pull requests.** Both passively and actively reviewing pull requests is a good idea before starting to work on a project because they represent the new, removed, or changed code that aren’t yet a part of the master branch. *Passively* reviewing pull requests means to read through the most recently opened and modified pull requests and see what kinds of contributions to the project are in the pipeline. This review helps to make sure that you don’t start working on something that someone else is either already working on or that may break or contradict something that someone else is already working on. If you detect a problem, you can add a comment to the discussion on the pull request.

  You can also *actively* review pull requests if have the proper permissions for the project (and, more importantly, if you're confident that you can evaluate whether the changes should be merged). You can start the review process by opening a pull request, clicking the Files changed tab, and then clicking the Review changes button at the top right of the pull request, shown in Figure 5-4. In the example for this book, no pull requests are open for our website. but we have one that is closed (the initial website template that we added).



Start a pull request review

thewecanzone / GitHubForDummiesReaders

Create neverAddMe.md #2

Open sguthals wants to merge 3 commits into `master` from `pr-review-practice`

```
5 ■■■■■ neverAddMe.md
... @@ -0,0 +1,5 @@
1 + This file should never be merged.
2 +
3 + The purpose of this document is to allow GitHub For
    Dummies reader to learn practice reviewing a pull
    request!
4 +
5 + I am proposing this file to be added.
```

Watch ▾ 0
Star 0
Fork 0
Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
Conversation 1
Commits 3
Checks 0
Files changed 1
Changes from all commits ▾
File filter... ▾
Jump to... ▾
+5 −0
Diff settings ▾
Review changes ▾
Copy path
View file

FIGURE 5-4: The place to start a review of a pull request.

TRIAGING ISSUES

Just like with an emergency room full of patients, triaging issues is the process of sorting and classifying issues on your repo. Okay, triaging issues may not be just like an emergency room because the issues are hopefully not life-threatening, but they're often critical for your project. If you're the owner of a repo on GitHub, we recommend spending the first 30 minutes of your day on a project triaging anything that may have come in since the last time you visited the repo. In the triage process, you should include at least

- **Apply labels.** Issues and pull requests can have labels associated with them. A few must-have labels are

bug: An issue that reports something that appears to be broken and should be addressed quickly

good first issue: An issue that a new contributor could start with

help wanted: An issue where the code owner is specifically asking for help from the community

needs investigation: An issue with questions that need to be answered before a solution is known

feature/enhancement: An issue that requests a new feature or change to the project

Some labels are added to a repo by default, including: bug, duplicate, enhancement, good first issue, help wanted, invalid, question, and wontfix. You can delete labels you don't want to use. Other labels can easily be added.

- **Respond to comments.** Some issues may have new comments since the last time you were on the repo. For example, if someone opened an issue and you applied the label `needs investigation` and asked for additional information from the person who opened the issue, you may want to check to see whether he or she provided you with any additional information first. Issues should remain active. If they go stale (as in folks stop commenting or making progress on them), they should probably just be closed.
- **Close stale issues.** Any issues that have gone stale should be closed. One example of when closing an issue is appropriate is when it's been a few weeks since the person who opened the issue last provided information or you know for a fact that you don't want to add a requested feature. Always comment on the issue before closing and let folks know why it was closed. These comments are also useful for your future self to remember why you closed an issue.

TIP

Chapter 11 covers the pull request review experience. However if you want to have an interactive experience where you get to review a pull request on your own project before doing it on someone else's project, you can visit `https://lab.github.com/githubtraining/reviewing-pull-requests`, authorize, accept Terms and Conditions, install GitHub Learning Labs, and take the automated, self-guided learning lab built by the great folks at GitHub. You can also practice reviewing a pull request on a repo designed specifically for this book, which you can find at `https://github.com/thewecanzone/GitHubForDummiesReaders/pull/2`.

Preparing Your Contribution

After you orient yourself with your project, you need to decide what you're going to work on for your contribution. For the example in this book, we choose issue #2 in the To do column of the project board: Change the title and tagline.

TIP Other good candidates are any issues that were opened with the label `bug`, `help wanted`, or `good first issue` because these issues are typically urgent for owners or good entry points for contributors.

This section assumes you already have a repo cloned onto your machine. The examples in this section use GitHub Desktop and Atom to resolve the issues. If you need guidance in cloning a repo or getting the project set up in GitHub Desktop or Atom, see Chapter 4.

Creating a branch for your contribution

Before you modify or add code to any project, whether a private solo project you own or a large open source project, we highly recommend creating a specific branch where all the code you write will be contained. Branching is a Git function that essentially copies code (each *branch* is a copy of the code), allows you to make changes on a specific copy, and then merge your changes back into the main (master) branch.

When you create a branch, Git doesn't actually copy all the code, which would be time consuming and inefficient. Git does something way smarter, but the specifics of what it does aren't important to day-to-day usage of Git, which is why we say Git creates a copy of the code. It's not technically correct, but it's conceptually correct. It's a useful mental model for branches.

Creating a branch for your changes gives you a safe space to try out solutions and make mistakes without threatening your project's integrity. If your solution is taking you down a wrong path, you can simply delete the branch, create a new one, and start over. Creating a branch helps your `master` branch remain in an always working state because you only merge your code into the master branch when you're confident it doesn't introduce any new problems and/or accurately solves the problem you were targeting.

Different projects and people use different nomenclatures for their branches. In this book, we use a short description of the code modifications to name our branches. The folks who contribute to Atom, which is an open source project hosted on GitHub, often put their initials in front of their branches before a short description. You can see this nomenclature on the Atom GitHub repo at `https://github.com/atom/atom/branches`. Before creating a branch on a public repo, check out how other contributors have been naming their branches or see whether the `CONTRIBUTING.md` file contains any guidelines.

REMEMBER This book uses GitHub Desktop to manage projects on your local machine and Atom as the primary text editor. If you need to set up these two applications, see Chapter 4.

To create the branch for your contribution, follow these steps:

1. **Verify the repo you cloned.**

   Open the GitHub Desktop application and verify that the top left drop-down list has your website repo selected. Figure 5-5 shows the repo correctly selected.

2. **Create a new branch.**

   Click the branch drop-down list in the top center of GitHub Desktop and click the New Branch button (see Figure 5-5). When you click the New Branch button, a dialog box appears.

3. **Give your branch a name.**

   Type a name for your branch. In our example, we use `new-title-and-tagline` because that is the change we plan to make.

4. **Click Create Branch.**

   The dialog box closes, and you switch to the new branch. Figure 5-5 shows the current branch as `master`, but when you complete this step, it should change to your new branch name. The button at the top right of GitHub Desktop also changes from Fetch origin to Publish branch.

5. **Publish your branch.**

   TIP Before making any changes to your branch, we recommend publishing your branch to GitHub.com. That way, as you start to modify or add code and push it to your branch, you won't lose any your work. Click the Publish branch button on the top right of GitHub Desktop. When the branch has been successfully published, this button goes back to the Fetch origin button, just like it is in Figure 5-5.

6. **Open your project in Atom.**

   Open the Atom application and verify that your project folder is open. A tree-view of your project appears in the left hand column. Notice that the branch selector at the bottom of the Atom application window now shows the branch name. Figure 5-6 shows the correct branch checked out in Atom.

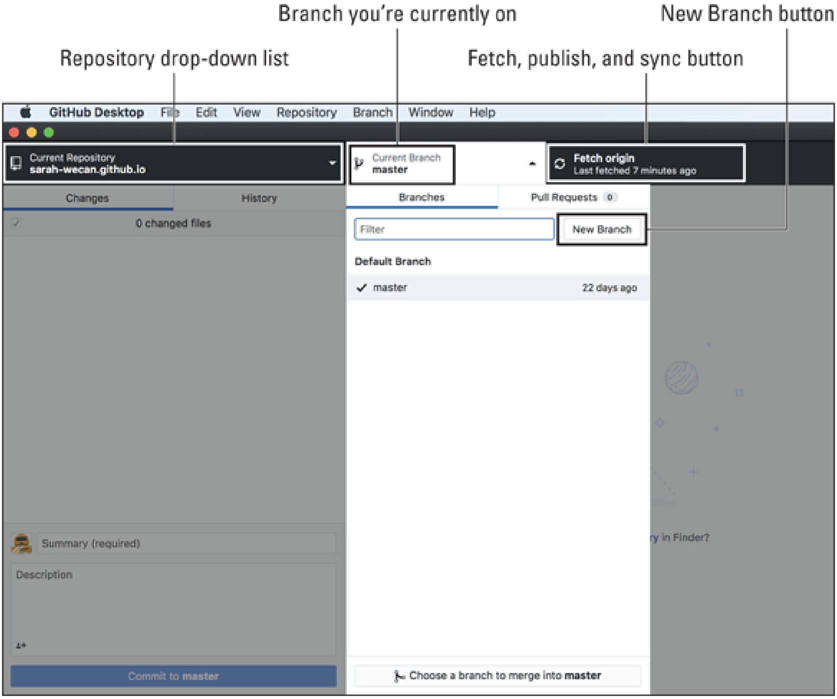
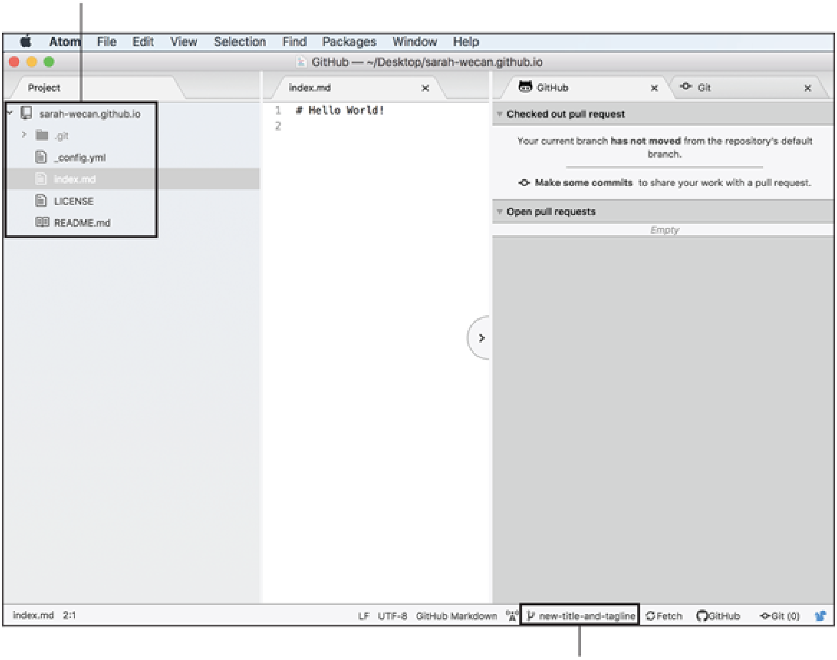

Branch you're currently on
New Branch button
Repository drop-down list
Fetch, publish, and sync button
GitHub Desktop
File
Edit
View
Repository
Branch
Window
Help
Current Repository
sarah-wecan.github.io
Current Branch
master
Fetch origin
Last fetched 7 minutes ago
Changes
History
0 changed files
Branches
Pull Requests 0
Filter
New Branch
Default Branch
master
22 days ago
Summary (required)
Description
Commit to master
Choose a branch to merge into master
ry in Finder?

FIGURE 5-5: The button to create a new branch in GitHub Desktop for a specific repo

File tree view


Atom File Edit View Selection Find Packages Window Help
GitHub — ~/Desktop/sarah-wecan.github.io
Project
sarah-wecan.github.io
.git
_config.yml
index.md
LICENSE
README.md
index.md
GitHub
Git
Checked out pull request
Open pull requests
Empty
index.md 2:1
LF UTF-8 GitHub Markdown
new-title-and-tagline
Fetch
GitHub
Git (0)

```
# Hello World!
```

Your current branch **has not moved** from the repository's default branch.

Make some commits to share your work with a pull request.

Currently checked out branch

FIGURE 5-6: The branch selector in Atom showing that the correct branch is checked out.

TIP If you click the branch selector in Atom and change the branch, the branch in GitHub Desktop correctly changes to the same branch, and vice versa!

Confirming your branch is published

Before writing code that you want to keep, confirm that you correctly published your branch and are able to push to the branch. If you're working on the same project, on the same computer, then you most likely are still properly set up. But if you're contributing to a project for the first time or you have just set up a new computer, you should consider confirming you're able to contribute.

This section shows you examples of contributing to the GitHub website repo `https://github.com/sarah-wecan/sarah-wecan.github.io` using GitHub Desktop and Atom.

To get started, follow these six steps:

1. **Click a file in the file tree in Atom.**

   Th file opens in the editor. Figure 5-7 shows `index.md` open.

2. **Modify the file.**

   TIP Don't make a lot of changes at this stage because the goal is to confirm you're able to push changes to the branch that you published.

3. **Save the file.**

   Choose File ⇒ Save to save your changes. When you click Save, The color of the file in the file tree changes, and the Git button changes to indicate that a change occurred (see Figure 5-7).

4. **Open the Git tab by clicking the Git button.**

   You see the changes in the unstaged area.

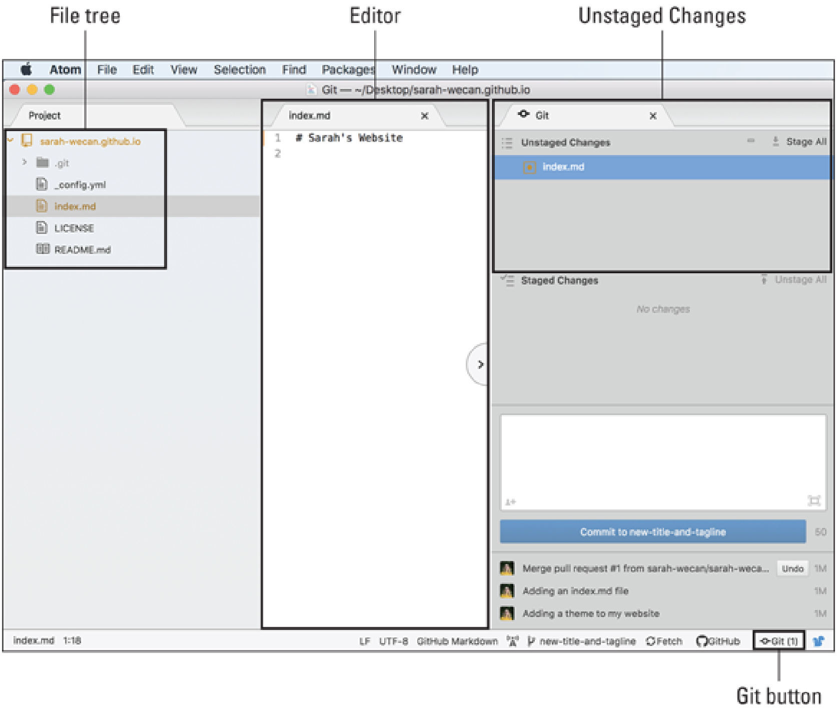
5. **Stage and commit the changes.**

   Stage all commits by clicking the Stage all button and write a commit message. Then click Commit to master. Figure 5-8 shows this step. After you click the Commit button, your commit appears in the commit list below the button, and a button that allows you to undo the commit appears.

6. **Push and merge the changes.**

   The Sync button on the bottom bar to the right of the branch name changes its action based on your current context. Prior to creating the commit in the previous step, the button was labeled Fetch (see Figure 5-8). It now is labeled Push 1 to indicate that you have one commit that you can push. Click Push 1 to push your changes to GitHub. Then open GitHub.com and go to the repository. In this example, the repository is `https://github.com/sarah-wecan/sarah-wecan.github.io`. You now see a suggestion to open a pull request (see Figure 5-9).

Create and merge the pull request. Chapter 3 gives a brief introduction to how to merge a pull request, and Chapter 8 gives an in-depth view of what you can do with pull requests if you need additional guidance.


File tree
Editor
Unstaged Changes
Atom
File
Edit
View
Selection
Find
Packages
Window
Help
Git — ~/Desktop/sarah-wecan.github.io
Project
sarah-wecan.github.io
.git
_config.yml
index.md
LICENSE
README.md
index.md
Git
Unstaged Changes
Stage All
index.md
Staged Changes
Unstage All
No changes
Commit to new-title-and-tagline
50
Undo
1M
1M
1M
index.md 1:18
LF
UTF-8
GitHub Markdown
new-title-and-tagline
Fetch
GitHub
Git (1)
Git button

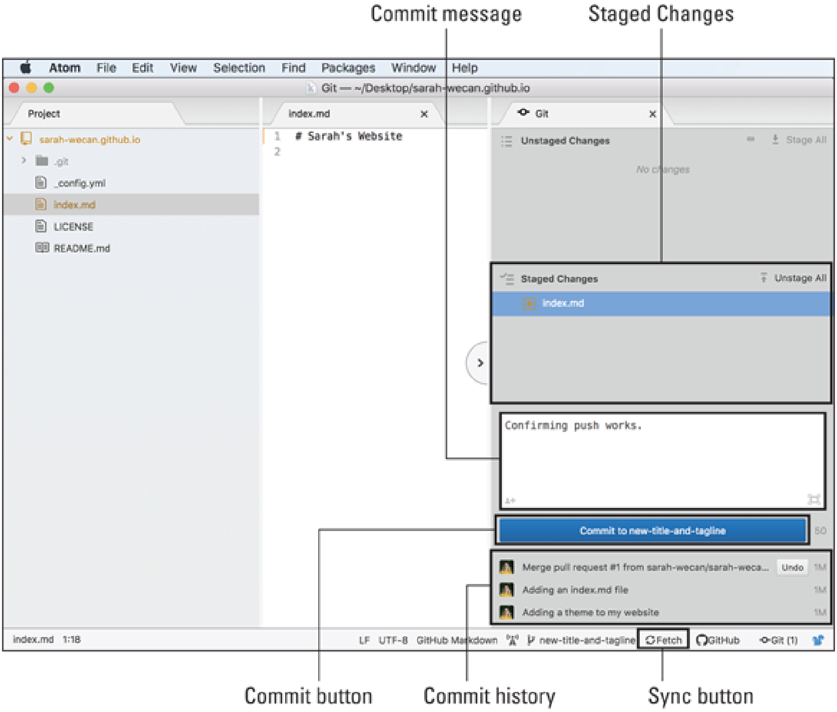
```
# Sarah's Website
```

Merge pull request #1 from sarah-wecan/sarah-weca...

Adding an index.md file

Adding a theme to my website

FIGURE 5-7: Editing a file in Atom.


Commit message
Staged Changes
Atom
File
Edit
View
Selection
Find
Packages
Window
Help
Git — ~/Desktop/sarah-wecan.github.io
Project
sarah-wecan.github.io
.git
_config.yml
index.md
LICENSE
README.md
index.md
Git
Unstaged Changes
Stage All
No changes
Staged Changes
Unstage All
index.md
Commit to new-title-and-tagline
50
Undo
index.md 1:18
LF
UTF-8
GitHub Markdown
new-title-and-tagline
Fetch
GitHub
Git (1)
Commit button
Commit history
Sync button

```
# Sarah's Website
```

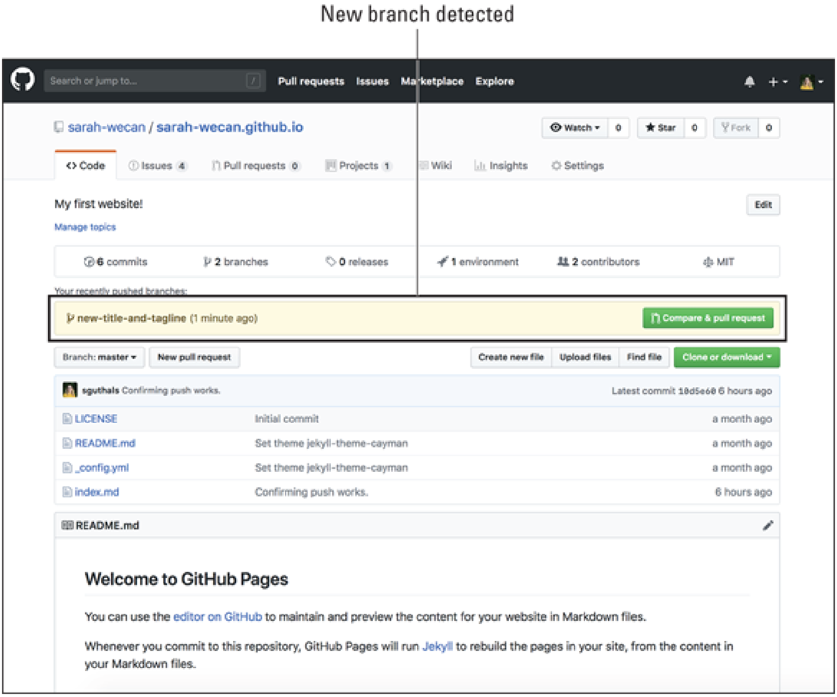
```
Confirming push works.
```

Merge pull request #1 from sarah-wecan/sarah-weca... 1M

Adding an index.md file 1M

Adding a theme to my website 1M

FIGURE 5-8: Staging and committing changes in Atom.


New branch detected
Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / sarah-wecan.github.io
Watch 0
Star 0
Fork 0
Code
Issues 4
Pull requests 0
Projects 1
Wiki
Insights
Settings
Edit
Manage topics
6 commits
2 branches
0 releases
1 environment
2 contributors
MIT
Your recently pushed branches:
new-title-and-tagline (1 minute ago)
Compare & pull request
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
sguthals Confirming push works.
Latest commit 10d5e60 6 hours ago
README.md

My first website!

LICENSE	Initial commit	a month ago
README.md	Set theme jekyll-theme-cayman	a month ago
_config.yml	Set theme jekyll-theme-cayman	a month ago
index.md	Confirming push works.	6 hours ago

Welcome to GitHub Pages

You can use the editor on GitHub to maintain and preview the content for your website in Markdown files.

Whenever you commit to this repository, GitHub Pages will run Jekyll to rebuild the pages in your site, from the content in your Markdown files.

FIGURE 5-9: GitHub.com indicates when a branch is ready to be merged.

Sometimes, pull requests can create merge conflicts. See the sidebar "Resolving simple merge conflicts" for insight on how to resolve them.

RESOLVING MERGE CONFLICTS

Sometimes your branch can get into a state where GitHub.com can't determine how to properly merge the code. This conflict can sometimes happen when a complex sequence of events happens. First, you create a branch from the `master` branch. As you make changes to your branch, someone else makes changes to the `master` branch. You then stage, commit, and push your changes to your branch, without pulling the changes made on the `master` branch into your branch first. The changes that you made may conflict with the changes that were made on the `master` branch — for example, you changed the same line, but started from a different place. For example, say that when you create your branch, the line is

```
Hello World!!!
```

and you changed it to

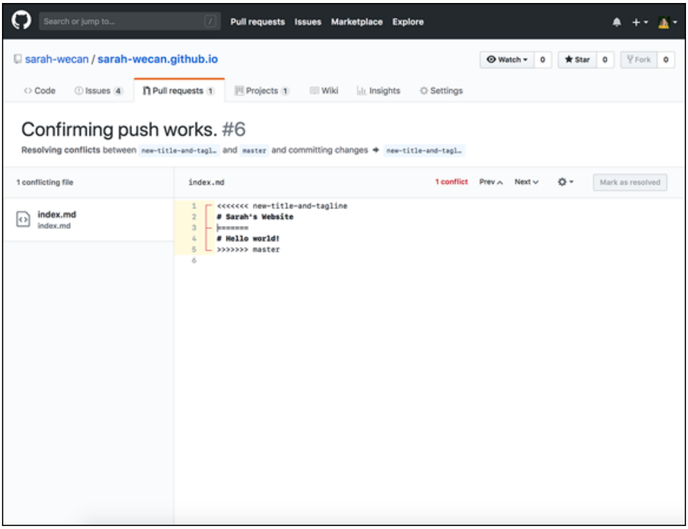
```
Sarah's Website
```

But someone else changed the line on the `master` branch to

```
Hello Friend!
```

When you create the pull request, GitHub gets confused. The commit shows that you changed code from `Hello World!!` to `Sarah's Website`, but the code that you're trying to merge says `Hello Friend!`, so it's unclear what should actually change. Rather than making a change for you that you don't intend to make, GitHub asks you what you want to do. When you create the pull request, instead of a button allowing you to merge the pull request, you are prompted with a button to resolve the conflicts. Clicking this button takes you to a new page that presents you with the line of code with two options (see figure):

- Code in your branch (on top)
- Code in the `master` branch (on bottom)


Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / sarah-wecan.github.io
Watch 0
Star 0
Fork 0
Code
Issues 4
Pull requests 1
Projects 1
Wiki
Insights
Settings
1 conflicting file
index.md
index.md
1 conflict
Prev
Next
Mark as resolved

Confirming push works. #6

Resolving conflicts between `new-title-and-tagl…` and `master` and committing changes → `new-title-and-tagl…`

index.md

```
<<<<<<< new-title-and-tagline
# Sarah's Website
=======
# Hello world!
>>>>>>> master
```

You can edit the code by deleting the delimiter lines and choosing the line you want to keep. In our example, the final code looks like

```
# Sarah's Website
```

Then, you can click Mark as resolved and commit the changes to your branch.

Building Your Personal Website

This section is following the example website repo that is created in Chapter 4 to show some basics on how to build a website. However, the tips shown in this section can be used for any website you build using GitHub Pages.

Modifying the title and tagline

To modify the title and tagline of your website, open the `_config.yml` file in Atom. Add two lines above the only line that is in the file, indicating the theme. Your code should look like this:

```
title: <Your Name>
description: <Your Description>
theme: jekyll-theme-cayman
```

Adding sections to your website

Adding sections to your website is made easy with Jekyll and GitHub Pages because you can use Markdown and HTML. First, include some social media usernames. Open your `_config.yml` file and add as many social media usernames as you want. In this example, we add a Twitter and GitHub username:

```
title: Your Name
description: Your Description
twitter_username: Your Twitter username
github_username: Your GitHub username
theme: jekyll-theme-cayman
```

```
Then, open the index.md file and change the code to include
   sections. For example, our code looks like this:# My
   Projects
Here are a list of projects that I am have been working on:

# My Interests
I'm interested in teaching novice coders about computer
   science!

# My Blog
I'm really excited to blog my journey on GitHub.com.

# Get in Touch
<ul>
```

```
<li><a href="https://twitter.com/{{ site.twitter_username
   }}">Twitter</a></li>
<li><a href="https://github.com/{{ site.github_username
   }}">GitHub</a></li>
</ul>
```

Save, stage, commit, and push your changes to your branch and then create and merge the pull request into the master branch. After a couple of minutes, refresh the website page to see your changes.

Creating a blog

Having a blog on your site is a great way for you to share your GitHub journey with others. As you start to discover and create, you can share what you learn and build with a community of people with similar interests. This section guides you through creating the blog posts and linking them from your `index.md` file.

First, create a new folder called `_layouts` and create a file within that folder called `post.html`. The `_layout/post.html` file should contain the following code to create a blog-style:

```
layout: default

<h1>{{ page.title }}</h1>
<p>{{ page.date | date_to_string }} - {{ page.author }}</p>

{{ content }}
```

WARNING Make sure that you correctly name the folder `_layouts`. Jekyll searches for the `_layouts` folder for any custom layouts. Otherwise, it uses the defaults for that theme. You can change the name of the specific layout — for example, `post.md` — but it should match the layout metadata, as shown in the following code snippet.

Using layouts and specific naming, Jekyll can extrapolate the title, date, and author to display that both on the blog post page and on the home page.

Then, create a new folder called `_posts` and a file inside with the date and a title. Typically, blog posts are made with `YEAR-MONTH-DAY-TITLE.md`. For example, this code is in a file called `_posts/2019-01-01-new-year.md`:

```
---
layout: post
author: sguthals
---
Write your blog post here.
```

Finally, in your `index.md` file, add the following code below the My Blog section:

```
<ul>
  {% for post in site.posts %}
    <li>
      <a href="{{ post.url }}">{{ post.title }}</a>
    </li>
  {% endfor %}
</ul>
```

Save, stage, commit, and push your changes to your branch and then create and merge the pull request into the master branch. After a couple of minutes, refresh the website page to see your changes.

Linking project repos

You can link GitHub project repos to your website in the same way you link social media, described in the section “Adding sections to your website,” earlier in this chapter. Putting a link directly to a repo can be efficient. However, you can also create web pages for project repos as well, as described in Chapter 4.

Open the `index.md` file and add the following code, replacing the URLs with links to projects you're the author of. This code shows linking to a project repo web page and directly to a project repo:

```
<ul>
<li><a href="https://sarah-wecan.github.io/HelloWorld/">Hello
    World Project</a></li>
<li><a href="https://github.com/thewecanzone/
    GitHubForDummiesReaders">GitHub For Dummies Repo</a></li>
</ul>
```

TIP

Out of scope for this book are a vast number of ways you can customize your GitHub Pages website. Jekyll and GitHub come together to offer a unique experience that requires some coding, but handles a lot of the setup. To find out how to do something specific, start by visiting GitHub Help at `https://help.github.com/categories/customizing-github-pages`.

Part 3

Contributing to Your First Project

IN THIS PART …

Fork your first GitHub repository so that you can contribute your own code.

Get unstuck when you've cloned and changed code before forking.

Create effective commit messages to communicate the changes you've made.

Create a pull request to start the process of your code being merged in.

Explore effective pull request workflows.

Review a pull request.

Chapter 6

Forking GitHub Repositories

IN THIS CHAPTER

- **Understanding forking**
- **Forking a repository**
- **Getting unstuck with forking and cloning**
- **Contributing code via a fork**

More than likely, you will want to work on some repositories where you are not the owner or collaborator. In situations where you aren't the owner or collaborator, you will have to fork the repo if you want to do anything other than browse the files.

In this chapter, we explain what forking is, show you how to fork a repository, and compare forking to cloning and duplicating. We also discuss contributing code via a fork. This chapter also demonstrates how to get the code you want to contribute into a fork if you've already made some changes to a clone before forking.

Introducing Forking

A *fork* of a repository is essentially just a copy of the repository. In the spirit of open source, forking is a way to share with and learn from other developers. Developers can have many motivations for forking a repository, but three of the most common reasons are to

- Contribute to someone else's project
- Use someone else's project as a starting point
- Experiment with someone else's code without making changes to their project

If you aren't the owner or a collaborator on a project, you'll have to fork the repo if you want to do anything other browse the files. If you plan on making any changes to a repo where you aren't an author or collaborator, fork the repo first so that you're in the correct state as you start to explore and modify the code. Don't worry, though, if you forgot to fork — see the section "Getting unstuck when cloning," later in this chapter, for help you.

TIP Prior to GitHub, a fork of an open source project tended to have a negative connotation. It wasn't just a copy of the source code, but a split in the community. A fork implies a fork in the road, where one group takes the project in a new direction. For example, Joomla is a fork of the Mambo project by a group of people that felt like a company had too much control. In practice, forks tend to be good for the overall ecosystem because they introduce new ideas. In some cases, the best ideas in the fork make their way back into the original project, such as when EGCS, which forked from GCC, had its changes merged back into the GCC project. On GitHub, forks tend to be more like short-lived branches that are either merged back into the main code or deleted.

Cloning, Forking, and Duplicating

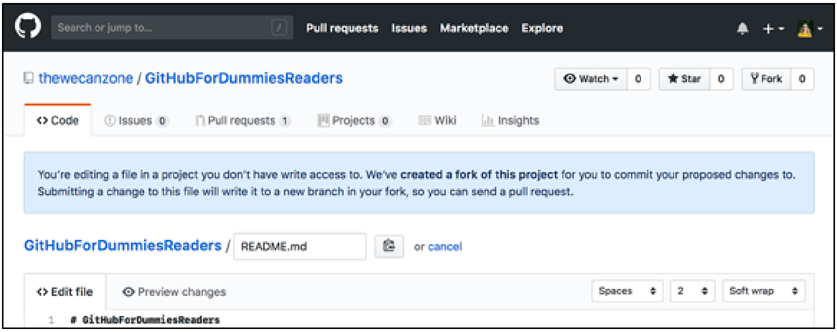
When you *clone* a GitHub repo, you're creating a local copy of the project on your computer. Forking a GitHub repo creates a copy of the repo on your GitHub.com account, and from there, you can clone the repo. A link between the original repo and the one you forked will remain, allowing you to pull changes made on the original repo into your copy and push changes that you make on your copy to the original copy.

Duplicating a repo is when you make a copy that no longer has a link to the original copy. Duplication isn't a usual part of an open source workflow because it makes it more difficult to push changes back into the original repository. Even so, duplicating a repo can be useful sometimes, such as when the original project is no longer active and you plan to keep the project alive with your fork.

Cloning a Repository

Any public repo can be cloned, and you can run the code on your computer and make changes to the code. But you won't be able to push those changes back to the remote repo if you don't have push permissions to the repo. Chapter 4 describes how to clone a repository in GitHub Desktop when you're the owner. The process is the same whether you're an owner, collaborator, or visitor.

Before you clone a repository, you should verify that you're able to push changes to it. The easiest way to verify this ability is to go to the repository home page. If you see a Settings tab on the right side of the home page, then you likely have push rights. If you don't, you likely have to fork the repo first. Alternatively, you can try to edit a file on GitHub.com, and if you get the message shown in Figure 6-1, then you don't have permission to directly contribute to the project.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
GitHubForDummiesReaders / README.md
or cancel
Edit file
Preview changes
Spaces
2
Soft wrap
# GitHubForDummiesReaders

You're editing a file in a project you don't have write access to. We've **created a fork of this project** for you to commit your proposed changes to. Submitting a change to this file will write it to a new branch in your fork, so you can send a pull request.

FIGURE 6-1: GitHub.com error message if you don't have edit permissions on a repo.

WARNING If you do end up cloning a repository where you don't have permission to contribute to it, you can end up making changes and not being able to push them. The section "Getting unstuck when cloning," later in this chapter, gives steps for how to get out of this state.

After you have a repository cloned on your local computer, you can view and modify metadata about it in your terminal. Open the terminal and go to a directory where you have a GitHub repository. If you need an example, clone `https://github.com/thewecanzone/`

`GitHubForDummiesReaders` by typing

```
$ git clone https://github.com/thewecanzone/
    GitHubForDummiesReaders
Cloning into 'GitHubForDummiesReaders'…
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 15 (delta 4), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (15/15), done.
$ cd GitHubForDummiesReaders
```

You can verify where the remote/target repo is with the following command:

```
$ git remote -v
originhttps://github.com/thewecanzone/
    GitHubForDummiesReaders.git (fetch)
originhttps://github.com/thewecanzone/
    GitHubForDummiesReaders.git (push)
```

If you cloned the same repo as we did, you see the exact same origin URLs for `fetch` and `push`. You should see that the remote repo is one owned by `thewecanzone` and not `sarah-wecan`. Alternatively, if you run the same command on a repo that you own, you should see your username. For example, if we run the command in the directory where I cloned my website repo that we create in Chapter 4, I would see

```
$ git remote -v
originhttps://github.com/sarah-wecan/sarah-wecan.github.io.git
    (fetch)
originhttps://github.com/sarah-wecan/sarah-wecan.github.io.git
    (push)
```

If you try using the command on a Git repo that doesn't have a remote origin (meaning it isn’t hosted on GitHub.com or any other remote place), you simply won’t get any information back. For example, in Chapter 1, we create a simple Git repo called `git-practice`. Running the command in that directory gives you nothing back:

```
$ git remote -v
```

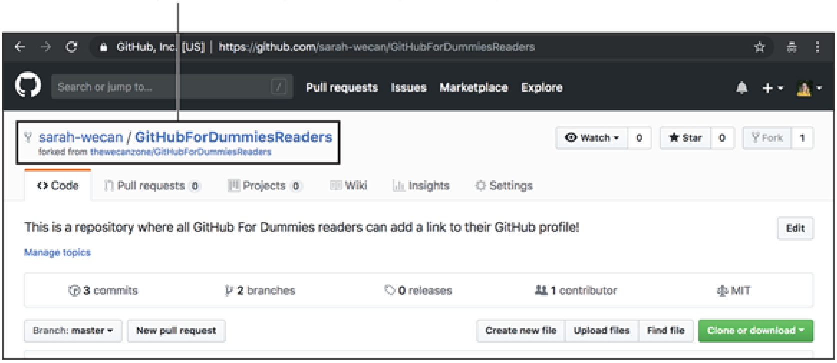
Forking a Repository

The goal of open source is to encourage collaboration among software developers around the world, so being able to contribute code to repositories where you aren't the owner or an explicit collaborator is an important part of the GitHub workflow and mission. To become a collaborator of an open source project, you can reach out to the owner of the repo and request to be a collaborator. However, if the owner doesn't know who you are, they probably won't add you as a collaborator because that would give you push rights to the repository. You'll have to gain their trust first.

REMEMBER You don't need the owner's permission to fork their repository. You can make your contributions and share them with the owner to show how you can be an asset to the project.

To fork a repo, go to the repo home page and click the Fork button at the top right. If you'd like, you can use `https://github.com/thewecanzone/GitHubForDummiesReaders` to practice forking and contributing to a public repo.

After you click the Fork button, the web page refreshes, and you see a slightly modified version of the repo, as shown in Figure 6-2. At the top of the repo, you see that the repo is attached to your account, but it still has a reference to the original repo.


Crumb trail linking the forked repo with the upstream repo

GitHub, Inc. [US] | https://github.com/sarah-wecan/GitHubForDummiesReaders
Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / GitHubForDummiesReaders
forked from thewecanzone/GitHubForDummiesReaders
Watch 0
Star 0
Fork 1
<> Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
Edit
Manage topics
3 commits
2 branches
0 releases
1 contributor
MIT
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

FIGURE 6-2: A forked repo on GitHub.com.

WARNING If you're part of multiple GitHub organizations, you're asked to choose which organization you want to fork the repository to after you click the Fork button and before the web page refreshes.

After you have your own, forked version of the repo, you can clone it on your local machine to start making changes. Chapter 7 goes over writing code and creating commits, which is the same process whether you're on a forked repo or a regular repo. If you clone the repo using GitHub Desktop, your local git repository knows about your forked version (remote `origin`) and the original repo (remote `upstream`).

TIP The concept of a *remote* can be confusing to those new to distributed version control systems like Git. When you clone a GitHub repository, you have a full copy of the repository on your local machine. You may be tempted to think the copy of the repository on GitHub is the canonical copy. However, there is no concept of canonical in Git. The *canonical copy* is whatever the people working on the project decide it is by consensus. Git does have the concept of a *remote,* which is a pointer to a copy of the same Git repository hosted elsewhere. Typically, a remote is a URL to a Git-hosting platform like GitHub, but it's possible to be a path to a directory with a copy of the repository. When you clone a repository, Git adds a remote named origin with the location (usually a URL) from where you cloned it. But it's possible to add multiple remotes to a Git repository to indicate other locations where you may want to push and pull changes from. For example, if you clone a fork of a repository, you may want to have a remote named upstream that points to the original repository.

If you clone the repo using the command line, you may want to set the upstream remote, which we explain in the section "Getting unstuck when cloning without forking," later in this chapter. You can see both the forked remote origin and original remote upstream if you run the `git remote -v` command in the directory where you cloned the repo:

```
$ git remote -v
originhttps://github.com/sarah-wecan/
   GitHubForDummiesReaders.git (fetch)
originhttps://github.com/sarah-wecan/
   GitHubForDummiesReaders.git (push)
upstreamhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (fetch)
upstreamhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (push)
```

The *origin,* which is where you fetch/pull changes from and push changes to, has your username (`sarah-wecan` in this example). The *upstream,* which is where the original code is located, and where you eventually want to contribute the code you write back to, has the original author's username (`thewecanzone` in this example).

Fetching changes from upstream

Having the upstream repo linked to your forked repo is important. As you start making changes, you want to be able to fetch/pull any changes that are being made on the original code into your code to make sure that you have the most up-to-date version.

For example, suppose that you forked and cloned a website project a week ago with plans to change the website's About page. While you were working on those changes, someone else made a change to the About page. Their changes may conflict with your changes, or they may introduce something new that you want to use in your changes. Pulling those changes into your local repository before you submit your changes back to the original repository makes sense. It reduces the chance that your changes conflict with the changes others are making to the About page and makes it more likely the owner can accept them.

If you find yourself in a situation where you need to get the change from the upstream, original repo, you can go to the directory where your forked repo is and type

```
$ git fetch upstream
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/thewecanzone/GitHubForDummiesReaders
   8404f3b..e02a4d2  master     -> upstream/master
$ git checkout -b new-branch
Switched to a new branch 'new-branch'
```

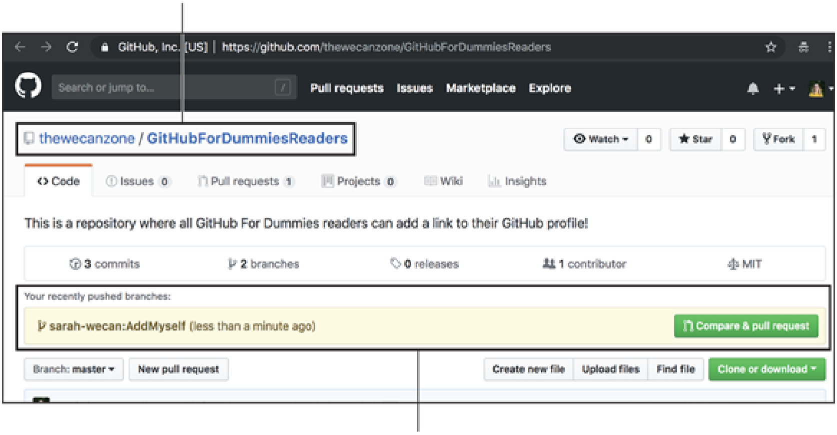
```
$ git merge upstream/master
Updating 8404f3b..e02a4d2
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

These three commands fetch the changes from the upstream repo, ensure that you're on your local, forked repo on a new branch, and then merges the changes from the upstream repo into your forked repo.

Contributing changes to upstream

After you make changes and publish them to a new in your forked repository, you're ready to suggest your changes to the original owner. If you go to the original, upstream repo on GitHub.com, your branch shows up on the home page, and GitHub asks whether you want to open a pull request to merge the changes with the original repo (see Figure 6-3).

Upstream repo name

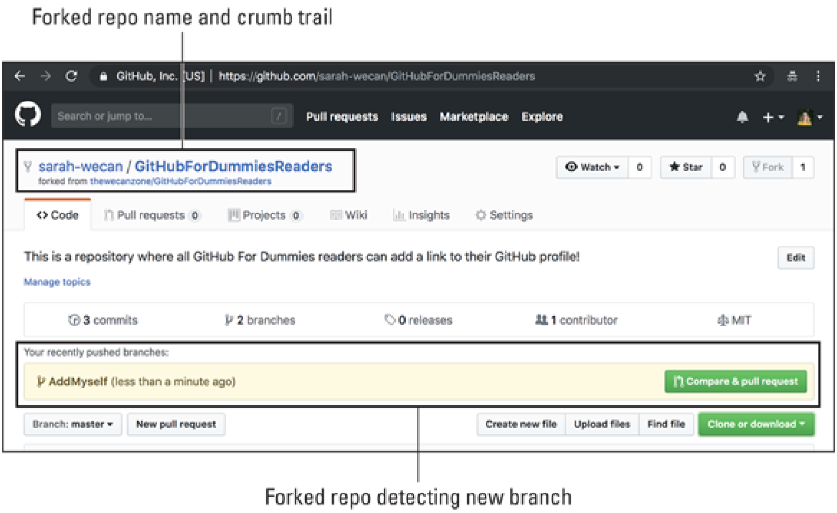

GitHub, Inc. [US] | https://github.com/thewecanzone/GitHubForDummiesReaders
Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
3 commits
2 branches
0 releases
1 contributor
MIT
Your recently pushed branches:
sarah-wecan:AddMyself (less than a minute ago)
Compare & pull request
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

Upstream repo detecting branch from forked repo

FIGURE 6-3: Original, upstream repo detecting a new branch from a forked repo.

On your forked repo on GitHub.com, your branch shows up, and GitHub asks whether you want to create a pull request for it (see Figure 6-4).

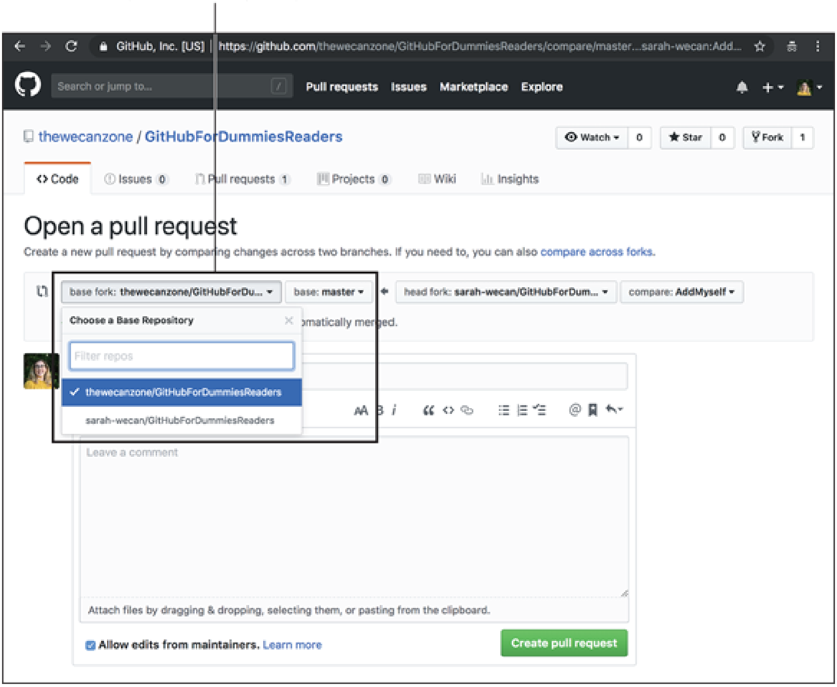


Forked repo name and crumb trail
GitHub, Inc. [US] | https://github.com/sarah-wecan/GitHubForDummiesReaders
Search or jump to...
Pull requests
Issues
Marketplace
Explore
sarah-wecan / GitHubForDummiesReaders
forked from thewecanzone/GitHubForDummiesReaders
Watch 0
Star 0
Fork 1
<> Code
Pull requests 0
Projects 0
Wiki
Insights
Settings
Edit
Manage topics
3 commits
2 branches
0 releases
1 contributor
MIT
Your recently pushed branches:
AddMyself (less than a minute ago)
Compare & pull request
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
Forked repo detecting new branch

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

FIGURE 6-4: Forked repo detecting a new branch.

Click the Compare & pull request button, and a pull request creation page gives you the option to request to merge your changes with the upstream repo or your forked repo (see Figure 6-5). Choose the upstream repo, add a comment, and click the Create pull request button.


Options for target repo

GitHub, Inc. [US] | https://github.com/thewecanzone/GitHubForDummiesReaders/compare/master...sarah-wecan:Add...

thewecanzone / GitHubForDummiesReaders

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: thewecanzone/GitHubForDu... base: master ← head fork: sarah-wecan/GitHubForDum... compare: AddMyself

Choose a Base Repository

✓ thewecanzone/GitHubForDummiesReaders

sarah-wecan/GitHubForDummiesReaders

...omatically merged.

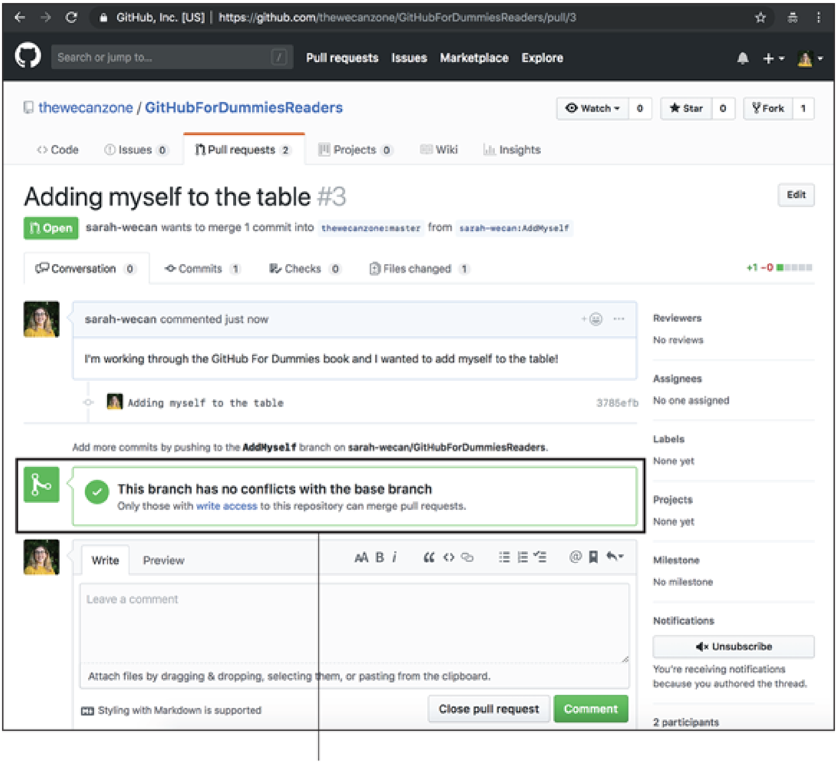
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

☑ Allow edits from maintainers. Learn more

Search or jump to...
Pull requests
Issues
Marketplace
Explore
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 1
Projects 0
Wiki
Insights
Filter repos
Leave a comment
Create pull request

FIGURE 6-5: Forked repo detecting a new pull request with the option for upstream or forked.

You see that the branch can be merged, but you have no way to personally merge the pull request is because you aren’t the owner of the target branch (upstream, original repo); only the owner (or a specified collaborator) has permission to merge code. Figure 6-6 shows the pull request on your repo without the option to merge.



Adding myself to the table #3

Open sarah-wecan wants to merge 1 commit into `thewecanzone:master` from `sarah-wecan:AddMyself`

sarah-wecan commented just now

I'm working through the GitHub For Dummies book and I wanted to add myself to the table!

`Adding myself to the table` 3785efb

Add more commits by pushing to the **AddMyself** branch on **sarah-wecan/GitHubForDummiesReaders**.

This branch has no conflicts with the base branch

Only those with write access to this repository can merge pull requests.

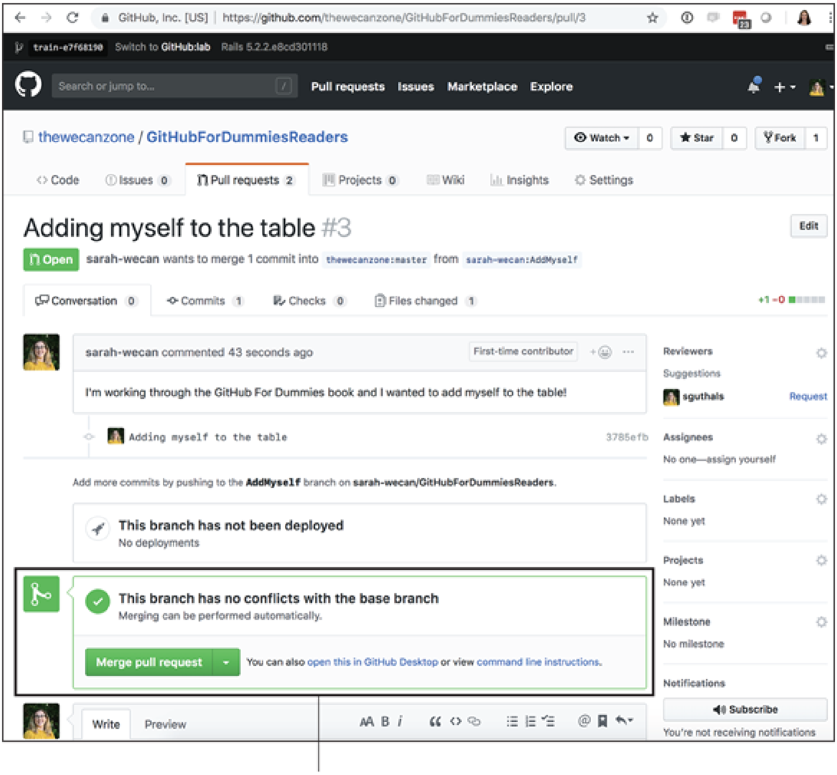
You're receiving notifications because you authored the thread.

GitHub, Inc. [US] | https://github.com/thewecanzone/GitHubForDummiesReaders/pull/3
Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 2
Projects 0
Wiki
Insights
Edit
Conversation 0
Commits 1
Checks 0
Files changed 1
+1 −0
Reviewers
No reviews
Assignees
No one assigned
Labels
None yet
Projects
None yet
Milestone
No milestone
Notifications
Unsubscribe
2 participants
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Close pull request
Comment

Unmergeable pull request status tile

FIGURE 6-6: Pull request without the option to merge.

As the owner of the upstream, original pull request, I can see the pull request and have the option to merge it (see Figure 6-7). If you're creating a pull request on this repo, I will continually merge pull requests so that we can keep an up-to-date table of all the *GitHub For Dummies* readers!


GitHub, Inc. [US] | https://github.com/thewecanzone/GitHubForDummiesReaders/pull/3
train-e7f68190 Switch to GitHub:lab Rails 5.2.2.e8cd301118
Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 0
Fork 1
Code
Issues 0
Pull requests 2
Projects 0
Wiki
Insights
Settings
Edit
Open
Conversation 0
Commits 1
Checks 0
Files changed 1
+1 −0
First-time contributor
Reviewers
Suggestions
sguthals
Request
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Notifications
Subscribe
Merge pull request
Write
Preview

Adding myself to the table #3

sarah-wecan wants to merge 1 commit into `thewecanzone:master` from `sarah-wecan:AddMyself`

sarah-wecan commented 43 seconds ago

I'm working through the GitHub For Dummies book and I wanted to add myself to the table!

`Adding myself to the table` 3785efb

Add more commits by pushing to the **AddMyself** branch on **sarah-wecan/GitHubForDummiesReaders**.

This branch has not been deployed
No deployments

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

You're not receiving notifications

Mergeable pull request status tile

FIGURE 6-7: Pull request with the option to merge.

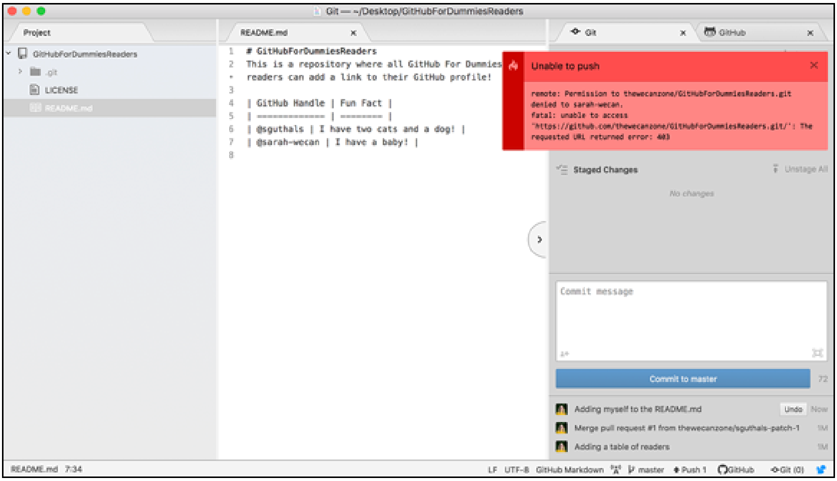
TIP If you have a lot of changes that you want to add to your fork before requesting that they get merged into the upstream, original repo, then you can first create the pull request to target your forked repo instead of the upstream repo. This is a change in what is shown in Figure 6-5. When you're ready to merge your changes into upstream, you can create a new pull request to request the target of the merge be the upstream repo.

Getting unstuck when cloning without forking

One common problem people run into is they forget to fork a repository before they try to contribute to it. The following scenario describes one example of getting into this situation.

Here’s the scenario: You clone a repository onto your local computer, modify the code, commit changes to `master`, and are ready to push your changes. But then you get a scary-looking error message. You may get the message in Atom (see Figure 6-8), GitHub Desktop (see Figure 6-9), or in the terminal:

```
$ git push origin master
remote: Permission to thewecanzone/GitHubForDummiesReaders.git
    denied to sarah-wecan.
fatal: unable to access 'https://github.com/thewecanzone/
    GitHubForDummiesReaders.git/': The requested URL returned
    error: 403
```



```
# GitHubForDummiesReaders
This is a repository where all GitHub For Dummies
readers can add a link to their GitHub profile!

| GitHub Handle | Fun Fact |
| ------------- | -------- |
| @sguthals | I have two cats and a dog! |
| @sarah-wecan | I have a baby! |
```

Unable to push

```
remote: Permission to thewecanzone/GitHubForDummiesReaders.git
denied to sarah-wecan.
fatal: unable to access
'https://github.com/thewecanzone/GitHubForDummiesReaders.git/': The
requested URL returned error: 403
```

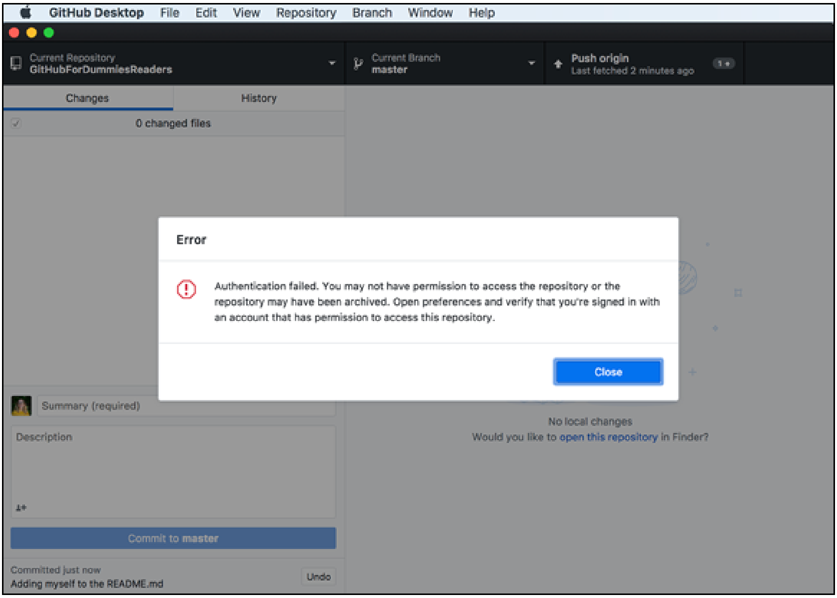
Adding myself to the README.md

Merge pull request #1 from thewecanzone/sguthals-patch-1

Adding a table of readers

Git — ~/Desktop/GitHubForDummiesReaders
Project
README.md
Git
GitHub
GitHubForDummiesReaders
.git
LICENSE
README.md
Staged Changes
Unstage All
No changes
Commit message
Commit to master
72
Undo
Now
1M
1M
README.md 7:34
LF
UTF-8
GitHub Markdown
master
Push 1
GitHub
Git (0)

FIGURE 6-8: Push permission error message in Atom.


GitHub Desktop
File
Edit
View
Repository
Branch
Window
Help
Current Repository
GitHubForDummiesReaders
Current Branch
master
Push origin
Last fetched 2 minutes ago
Changes
History
0 changed files
Summary (required)
Description
Commit to master
Committed just now
Adding myself to the README.md
Undo
Close

Error

Authentication failed. You may not have permission to access the repository or the repository may have been archived. Open preferences and verify that you're signed in with an account that has permission to access this repository.

No local changes

Would you like to open this repository in Finder?

FIGURE 6-9: Push permission error message in GitHub Desktop.

The error message tells you that you don't have permission to push to this repository. You should have forked the repository first. You also made the mistake of committing directly to master. As we recommend elsewhere in the book, it's a good practice to make all your changes in a branch.

To fix this mistake, you need to move your changes to a new branch, fork the repo, change the remote URLs for your local repository to point to your fork, and push your changes. This process can get tricky, but these steps can help you out of this predicament:

1. **Migrate your changes to a new branch.**

   Right when you discover you're targeting the incorrect remote repository, you should move your changes to a new branch. You don't want to accidentally pull in changes from the upstream, original branch onto all the hard work you just finished. This step can get tricky, but luckily Phil has created a Git Alias to help. See the nearby sidebar "Creating a Git Alias" for help. After you have the `git migrate` alias, go to the directory where your repo is in your terminal and type

```
$ git migrate new-branch
Switched to a new branch 'new-branch'
Branch 'master' set up to track remote branch
   'master' from 'origin'.
Current branch new-branch is up to date.
```

Confirm that the new branch has been created:

```
$ git status
On branch new-branch
nothing to commit, working tree clean
```

TIP You can also confirm that your commits are only in this new branch and no longer in the old branch by running a log command to compare the two branches:

```
$ git log master..new-branch --oneline
```

This lists the commits in `new-branch` that are not in `master`. The `--oneline` flag prints each commit on a single line, which is useful when you just need a summary of commits and not the full details.

2. **Set the upstream remote to be the original repo.**

   To add an upstream remote to your repo, go to the terminal and type

```
$ git remote add upstream https://github.com/
   thewecanzone/GitHubForDummiesReaders.git
```

   Confirm that the upstream remote was added correctly:

```
$ git remote -v
```

```
originhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (fetch)
originhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (push)
upstreamhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (fetch)
upstreamhttps://github.com/thewecanzone/
   GitHubForDummiesReaders.git (push)
```

3. **Fork the repo.**

   Back on GitHub.com, go to the original repo and click Fork at the top right of the repo home page. The page refreshes, and you see your own version of the repo, referencing the original repo (refer to Figure 6-2, earlier in the chapter).

4. **Set the origin remote to be your forked repo.**

   After you have your own fork of the repo, you can change your remote origin to be your version:

   ```
   $ git remote set-url origin https://
      github.com/sarah-wecan/
      GitHubForDummiesReaders.git
   ```

   You can also confirm that all your remote URLs are correctly set:

   ```
   $ git remote -v
   originhttps://github.com/sarah-wecan/
      GitHubForDummiesReaders.git (fetch)
   originhttps://github.com/sarah-wecan/
      GitHubForDummiesReaders.git (push)
   upstreamhttps://github.com/thewecanzone/
      GitHubForDummiesReaders.git (fetch)
   upstreamhttps://github.com/thewecanzone/
      GitHubForDummiesReaders.git (push)
   ```

5. **Push your branch to your forked version.**

   You're now in the same state that you would be in had you forked the repo before cloning. Back in Atom, you can publish your branch.

6. **Create a pull request.**

   And just like in Figure 6-4, earlier in this chapter, your forked repo detects a new branch and offers to have you create a pull request.

CREATING A GIT ALIAS

A *Git Alias* is an easy way to automate and extend Git commands. If you're doing a lot of Git commands on the terminal, creating Git Aliases can make your software development more efficient. For example, in your terminal you can type

```
$ git config --global alias.st status
```

Now, instead of typing `git status`, you can type `git st`, and Git returns the current status of your repository. Getting rid of just four letters may seem a little silly, but it can end up making your Git command experience a lot more efficient over time.

A Git Alias is a lot more powerful than just reducing the number of keys you have to press. Phil describes a tricky scenario in his blog post (`https://haacked.com/archive/2015/06/29/git-migrate`) where you have to migrate the commits you've made on a branch to another branch. This migration is critical if you get stuck in the position where you've started working on a clone of a repository where you don't have write permissions, as we discuss in the section "Getting unstuck when cloning without forking," earlier in this chapter.

The Git Alias to migrate commits from one branch to another is complex. It is a few complicated steps all rolled into one simple `git migrate` command. To make this command accessible for you to use when you get stuck on an unforked clone, follow these steps in your terminal:

```
$ open ~/.gitconfig
$
```

Your `.gitconfig` file opens in your default editor. Add the following code to the bottom of your `.gitconfig` file:

```
[alias]
migrate = "!f(){ CURRENT=$(git symbolic-ref --short
    HEAD); git checkout -b $1 && git branch --force
    $CURRENT ${3-$CURRENT@{u}} && git rebase --onto ${2-
    master} $CURRENT; }; f"
```

If your `.gitconfig` file already has an [alias] section, don't retype that line. Save and close the `.gitconfig` file.

Now you can use the `git migrate` command to migrate commits from one branch to another branch! This Git command has one required parameter and two optional parameters:

```
git migrate <new-branch-name> <target-branch> <commit-range>
```

The parameter `<new-branch-name>` is required. This branch is where you'll move the commits to. If you don’t specify anything else, then the migrate command will move all commits from the `master branch` to this new branch.

The parameters `<target-branch>` and `<commit-range>` are optional. `<target-branch>` allows you to move commits from a branch other than the `master branch` to the `<new-branch-name>` that you specify in the first parameter. `<commit-range>` allows you to specify which commits you want to move over. This parameter can be useful if you accidentally made one commit on the wrong branch, and you just want to move that one commit over to `<new-branch-name>.`

Chapter 7

Writing and Committing Code

IN THIS CHAPTER

- **Committing code in a terminal**
- **Creating a good commit**
- **Writing a commit message**
- **Committing other tools**

In this chapter, you write and commit code. The first part, writing code, is a very broad topic — too broad to be covered in this (or any single) book. The code we write in this chapter sets the stage for covering how to create good commits. Most of this chapter focuses on committing code. No matter what kind of code you write, the act of committing that code remains the same.

The code example we use throughout this chapter may seem contrived and overly simplistic. That's because it is contrived and simple. Don't let the simplicity, though, distract you because the information in this chapter also applies to large code bases.

Creating a Repository

A *commit* is the smallest unit of work with Git. It represents a small logical group of related changes to the repository. A commit additionally represents a snapshot in time — the state of the entire repository can be represented by referencing a single commit.

Before writing code, you need to create a local repository to store the code. In the following examples, we create a repository in a directory named `best-example`. Feel free to change `best-example` to a directory of your choice. Fortunately, this process is quick and painless:

1. **Open the terminal on your computer.**

   If you don't know how to do so, see Chapter 1 for guidance.

2. **Go to the directory where you want your project folder to be stored and type the following commands:**

   ```
   $ git init best-example
   $ cd best-example
   ```

   The first command creates an empty Git repository in the specified directory, `best-example`. Because the `best-example` directory doesn't already exist, Git creates it. The second command changes the current directory to this new directory.

TIP Nearly every Git tutorial I've seen that covers initializing a Git repository does it in the current directory by calling `git init` with no parameters or `git init .` where the `.` represents the current directory. People can be forgiven for not realizing you can both create the repository directory and initialize it in one step by passing in the path to the new repository like we do here. In fact, you can combine both of these commands into a single command: `git init best-example && cd best-example`. This tip can help you gain the admiration and adulation of your less efficient peers!

Writing Code

After you're in a Git repository directory, you can start adding files. (If you aren't in a directory, see the previous section, "Creating a Repository" where we created the `best-example` directory.)

For this example, you create three files by typing the following code:

```
$ touch README.md
$ touch index.html
$ mkdir js
$ touch js/script.js
```

Note that one of the files you create is a `README.md` file. To find out why every repository should have a `README.md` file, see Chapter 10.

After running these commands, you have three files:

- `README.md`
- `index.html`
- `script.js`

`script.js` is in a subdirectory named `js`. You guessed it. — you're making a simple website!

You can flesh out the `README.md` file first. In this example, we use Atom to open and edit the files in the current directory. (If you need any guidance setting up Atom, see Chapter 2.)

TIP If you prefer, you can easily use another editor, such as Visual Studio Code, to edit the file by replacing `atom` with `code` in the following example:

```
$ atom .
```

You can add some simple Markdown text to the `README.md` document. *Markdown* is language that offers a simple way to format and style your text. You can check out a guide on Markdown on the GitHub guides `https://guides.github.com/features/mastering-markdown`.

Open the`README.md` in the editor by clicking in in the file tree in Atom. Then add some Markdown relevant to your project. In this example, add the following text:

```
# The Best Example Ever


Which will be a part of the best commit ever.
```

Then add the following code to `index.html`.

```
<!doctype html>
<html lang="en">
<head>
```

```
  <meta charset="utf-8">
  <title>It is the cod3z</title>
  <script src="js/script.js"></script>
</head>
<body>
  <h1>The Best Cod3z!</h1>
</body>
</html>
```

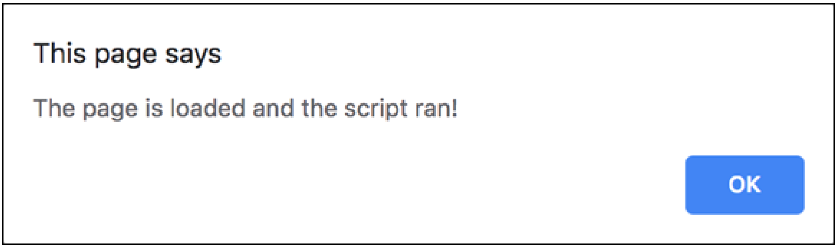
This HTML file references `scripts.js`. Open `script.js` in Atom and add the following code.

```
document.addEventListener(
  "DOMContentLoaded",
  function(event) {
    alert('The page is loaded and the script ran!')
  }
);
```

REMEMBER Make sure to save your changes to each file. Now test the code by opening `index.html` in your browser from the terminal.

```
$ open index.html
```

The page loads in your default browser, and the alert message, shown in Figure 7-1, appears.



This page says

The page is loaded and the script ran!

FIGURE 7-1: An alert message from our running code.

Creating a Commit

This section assumes you have code that you've changed on your local computer and that the code is in a working state. If you need an example of working code, see the previous section in this chapter to get to this state.

After you have running code, you can commit it to the repository. To create a commit is a two-step process:

1. **Stage the changes you want to commit.**
2. **Create the commit with a commit message.**

Staging changes

Staging changes can be confusing to the Git beginner. In concept, it’s similar to a staging environment for a website. Staging changes is an intermediate place where you can see the changes you’re about to commit before you commit them.

Why would you want to stage changes before committing them? In Git, a commit should contain a group of related changes. In fact, Git encourages this setup.

Suppose that you’ve been working for a few hours and now have a large set of unrelated changes that aren’t committed to the Git repository.

You may be tempted to just commit everything with some generic commit message like “A bunch of changes.” In fact, an XKCD commit makes light of this phenomena at `https://xkcd.com/1296`.

Committing a bunch of unrelated changes is generally a bad idea. The commit history of a repository tells the story of how a project changes over time. Each commit should represent a distinct cohesive set of changes. This approach to commits isn’t just about being fastidious and organized. Having a clean Git history has concrete benefits.

TIP One benefit of a clean Git history is that a command like `git bisect` is way more useful when each commit is a logical unit of work. The `git bisect` command is an advanced command, and full coverage of what it does is beyond the scope of this book. In short, though, `git bisect` provides a way to conduct a binary search through

> your Git history in order to find the specific commit that introduces a particular behavior, such as a bug. If every commit contains a large group of unrelated changes, finding the specific commit that introduces a bug isn't as useful as it would be if every commit contains a single logical unit of change.

In the example for this chapter, we can probably stand to create two commits:

- One that just contains the `README.md` file.
- Another that contains the `index.html` and `script.js` files.

Because the `index.html` file references the `script.js` file, checking in one without the other doesn't make sense at this point.

Start by staging the `README.md` file:

```
$ git add README.md
```

The `README.md` file is added to the Git index. The Git index is the staging area for creating commits to the repository. You can check the status of the repository to see that the file has been added to the index:

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>…" to unstage)

new file:   README.md

Untracked files:
  (use "git add <file>…" to include in what will be committed)

index.html
js/
```

As you can see, the `README.md` file is staged for commit. Meanwhile, the `index.html` and `js/` directory aren't yet tracked by this repository.

TIP Why isn’t `script.js` listed in the untracked files section? Git is taking a shortcut here. It notices that no files within the `js` directory are tracked, so it can simply list the directory rather than list every file in the directory. In a larger code base, you'll be glad Git isn’t listing every file in every subdirectory.

Committing a file

After you stage changes (see preceding section), you can create a commit. In this example, we use the `-m` flag with the `git commit` command to specify a short commit message. The following commands demonstrate how to create a commit and specify the commit message in one step:

```
$ git commit -m "Add a descriptive README file"
[master (root-commit) 8436866] Add a descriptive README file
 1 file changed, 3 insertions(+), 0 deletions(-)
create mode 100644 README.md
```

The file is committed. If you run the `git status` command again, you see that you still have untracked files. The `git commit` command commits only the changes that are staged.

Committing multiple file:

After you commit the first file, you're ready to stage the rest of the files for a commit.

```
$ git add -A
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>…" to unstage)

new file:   index.html
new file:   js/scripts.js
```

The `-A` flag indicates that you want to add all changes in the working directory to the Git index. When you run the `git status` command, you can see that you've staged two files.

TIP When the `js` directory is untracked, `git status` lists only the `js` directory and none of the files in the directory. Now that you're trying to stage the `js` directory, Git lists the file in the `js` directory. Why the discrepancy? Git doesn't actually track directories. It tracks only files. Therefore, when you add a directory to a Git repository, it needs to add each file to the index.

Sometimes you need to write a more detailed commit message. In this example, we don’t specify a commit message when we run the commit command because we plan to write a more detailed commit message:

```
$ git commit
```

If you don’t specify a commit message using the -m flag, Git launches an editor to create a commit message. If you haven’t configured an editor with Git, it uses the system default editor, typically VI or VIM.

There are legions of jokes about how difficult it is to exit VIM, so we won’t rehash them all here. We’ll simply take a moment of silence in remembrance for our friends still stuck in the VIM editor.

TIP For the record, to exit VIM, press the ESC key to exit the edit mode and type `:wq` to exit and save or `:q!` to exit without saving.

To change the default editor to something like Atom, run the following command in the terminal:

```
git config --global core.editor "atom --wait"
```

TIP The editor opens a temporary file named `COMMIT_EDITMSG`, which contains some instructions that are commented out:

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
# new file:   README.md
#
# Untracked files:
# index.html
# js/
#
```

You will enter your commit message in the file that gets opened. You can write your message before all the comments or simply replace everything in the file with your own commit message.

In this case, we replace everything in that file with

```
Add index.html and script.js

This adds index.html to the project. This file is the
default page when visiting the website.

This file references js/script.js, which is also added
in this commit.
```

After you save the commit message and close the file or editor, Git creates a commit with the message you wrote.

Writing a Good Commit Message

What should you write in a commit message? What makes a good commit message?

A Git commit should contain a logical and cohesive change or set of changes. The message should describe that change in clear terms so that anyone who reads the message later understands what changed in the commit.

REMEMBER The audience for the commit message are current and future collaborators on the project. Those collaborators may include yourself in the future. Someday you may be tracking down a bug and want to understand why you made some change. You'll thank past-you for writing a well-written commit message that answers that question. So write clear commit messages and be nice to future-you.

If you find that you have trouble describing a commit, it may be that the commit contains too many changes. In writing code, we often note that well written functions do one thing and do it well. Similarly, a commit should represent one change to the system. The commit message describes the change and why it’s being made.

A good commit message should also follow a specific structure. In general, a commit message has two parts:

- The **summary** should be short (50 characters or less) and in the imperative present tense. For example, instead of writing “I added a method to Frobnicate widgets,” write “Add method that Frobnicates widgets.”
- The **description** provides detailed explanatory text, if needed. Not every change requires explanatory text. For example, if you rename a function, a commit message with just a summary of “Rename Frobnicate to Bublecate” may suffice. You should wrap the description text at 72 characters. This ensures it looks good when displayed in the terminal as part of the output from the `git log` command.

By convention, a new line character separates the summary from the description.

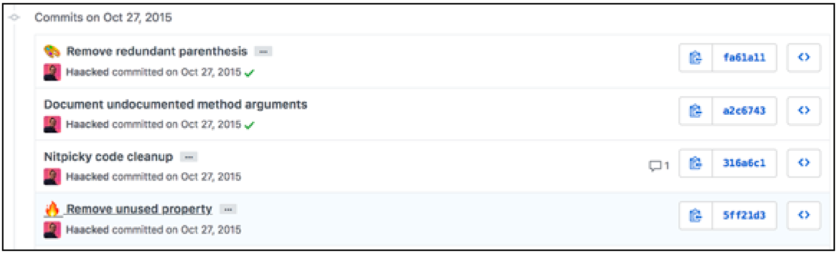
Here’s an example commit message written by one of the authors in an open source project `https://git.io/fhZ5a`:

```
Avoid potential race condition

In theory, if "ClearFormCache" is called after we
check `contains` but before we execute the `return`
line, we could get an exception here.
```

```
If we're concerned about performance here, we could
consider switching to the ConcurrentDictionary.
```

There’re a few conventions you can use within a Git commit message that Git will ignore, but GitHub will recognize. For example, you can specify that a commit resolves a specific issue with something like “fixes #123” where `123` is the issue number. When a commit with this pattern is pushed to GitHub, the issue number is linked to the issue. And when the branch that contains that commit is merged into the default branch of the repository (typically `master`), GitHub closes the referenced issue. That's pretty handy!

TIP You can also use emojis in a commit message. For example, one pattern some teams use is to indicate that a commit contains only cosmetic changes by prefacing it with `:art:`. When that commit is rendered on GitHub.com, GitHub renders the emoji. You can see this in action in Figure 7-2 which shows a list of commits from the GitHub for Visual Studio open source project `https://git.io/fhnDu`.



Commits on Oct 27, 2015

🎨 Remove redundant parenthesis …
Haacked committed on Oct 27, 2015 ✓
fa61a11

Document undocumented method arguments
Haacked committed on Oct 27, 2015 ✓
a2c6743

Nitpicky code cleanup …
Haacked committed on Oct 27, 2015
1
316a6c1

🔥 Remove unused property …
Haacked committed on Oct 27, 2015
5ff21d3

FIGURE 7-2: A list of commit messages, some with emojis in the summary.

Committing Code with GitHub Desktop

Even though committing from the terminal is pretty straightforward, many people prefer to use a GUI application to commit code. Using a GU has a few benefits:

- A GUI can provide guidance on conventions with commit messages, such as keeping the summary to 50 characters and separating it from the description by new lines. A GUI can simply present two fields: summary and description.
- A GUI can provide support for GitHub specific conventions, such as the one where you can specify that a commit resolves an issue.

GitHub Desktop (which we refer to as Desktop for short) is a GUI created by GitHub that is great for committing code.

Tracking a repository in Desktop

Choose a repository that you have never opened in Desktop, but that you have locally on your computer. (See Chapter 2 if you haven't worked with Desktop yet.) If you need an example, use the `best-example` repository that you can create in the section "Creating a Repository," earlier in this chapter. When you launch Desktop, the `best-example` repository isn't listed in the list of repositories. Desktop doesn't scan your computer for Git repositories to manage. Instead, you have to tell Desktop about each repository you want to manage.

As expected, if you use Desktop to clone or create the repository, it's already tracking it. But sometimes you have a repository that you cloned or created outside of Desktop — for example, we created `best-example` using the terminal. Now you need to tell Desktop to track the repository you have chosen. Fortunately, this task is easy from the terminal.

The Desktop command line tool allows you to launch Desktop from your terminal, which allows you to easily integrate Desktop as much or as little as you want into your existing terminal-based Git workflow.

On Windows, you don't need to install the command line tool; it's done automatically. On the Mac, you have to take a separate step.

To install the command line tool on a Mac:

1. **Make sure Desktop is the active application and then, in the application menu bar, choose GitHub Desktop ⇒ Install Command Line Tool.**
2. **From the terminal, make sure that you're in the repository you want Desktop to track.**

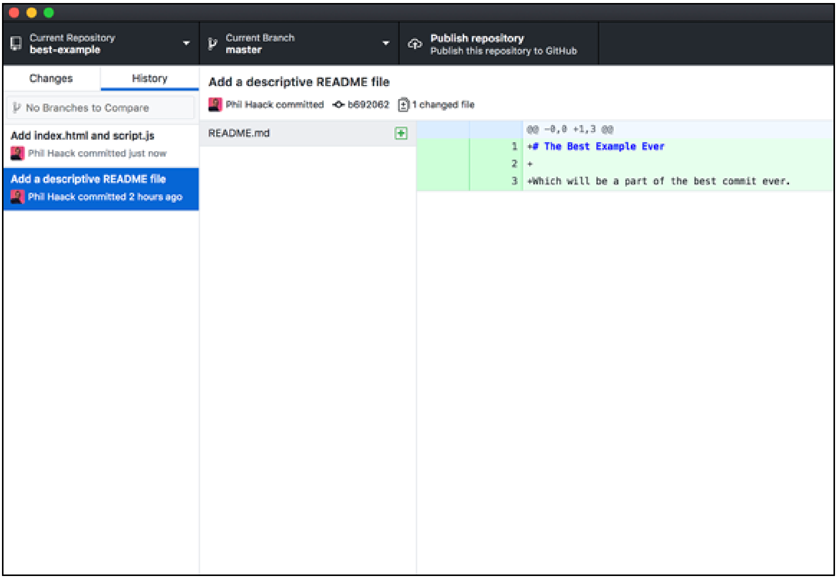
   For this example, we are in the `best-example`.

3. **Run the following command:**

   ```
   $ github .
   ```

   The `.` in the command represents the current directory. It could, instead, be a fully qualified path to a directory.

   GitHub Desktop launches (if it's not already running) and opens the specified directory. Because the current directory is already a Git repository, Desktop adds it to the list of repositories that it tracks. It then sets this repository as the current repository so that you can browse the repository's history, switch branches, and create commits, as shown in Figure 7-3.


Current Repository
best-example
Current Branch
master
Publish repository
Publish this repository to GitHub
Changes
History
No Branches to Compare
README.md

Add index.html and script.js
Phil Haack committed just now

Add a descriptive README file
Phil Haack committed 2 hours ago

Add a descriptive README file
Phil Haack committed b692062 1 changed file

```
@@ -0,0 +1,3 @@
+# The Best Example Ever
+
+Which will be a part of the best commit ever.
```

FIGURE 7-3: GitHub Desktop opened to the `best-example` repository.

If the current directory wasn't a repository, Desktop prompts you to create a Git repository in that directory. How convenient!

Publishing a repository in Desktop

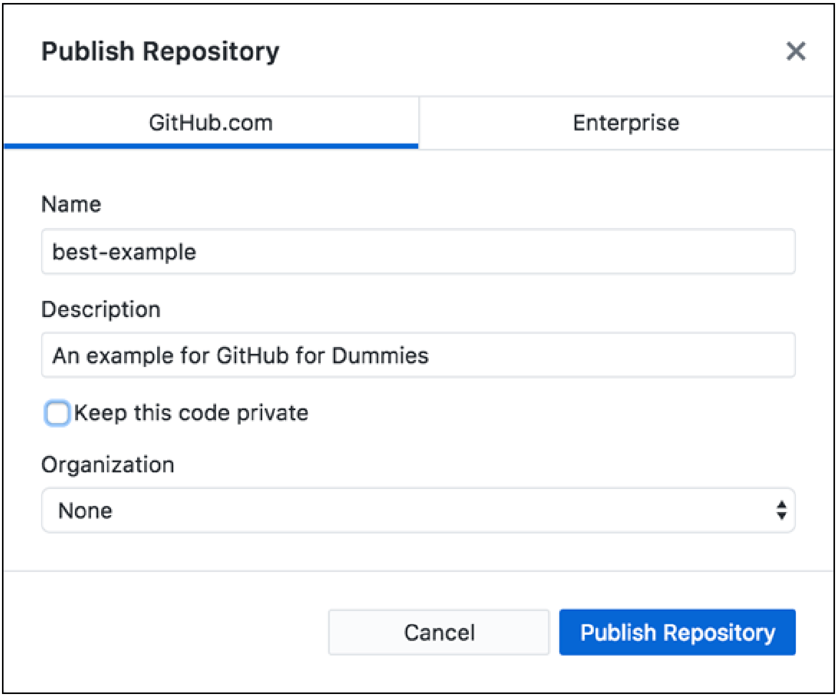
To experience the full power of Desktop's integration with GitHub, you need to publish this repository to GitHub:

1. **Clicking the Publish repository button.**

   A dialog box to publish the repository appears (see Figure 7-4).

2. **Fill in the details and click the Publish Repository button.**

   The repository is created on your GitHub.com account.


Publish Repository
×
GitHub.com
Enterprise
Cancel
Publish Repository

Name: best-example

Description: An example for GitHub for Dummies

☐ Keep this code private

Organization: None

FIGURE 7-4: The Publish dialog box used to publish a repository to GitHub.com.

TIP Desktop provides a keyboard shortcut to open the browser to the repository: ⌘ + Shift + G (Ctrl+Shift+G on Windows).

If you want, you can create a few issues in the repository. (see Chapter 3 to find out how to create issues.) For this example, we create five issues:

- Provide more details on the README.
- Mention in the README that this is a collaborative efforts.
- Add a contribution section to README.
- Do not use an alert message.
- Set up website alerts.

You can also see these issues on our repo at `https://github.com/FakeHaacked/best-example/issues`.

Committing in Desktop

Desktop is used only for Git operations. To edit the files in the repository, you still need to use your editor of choice.

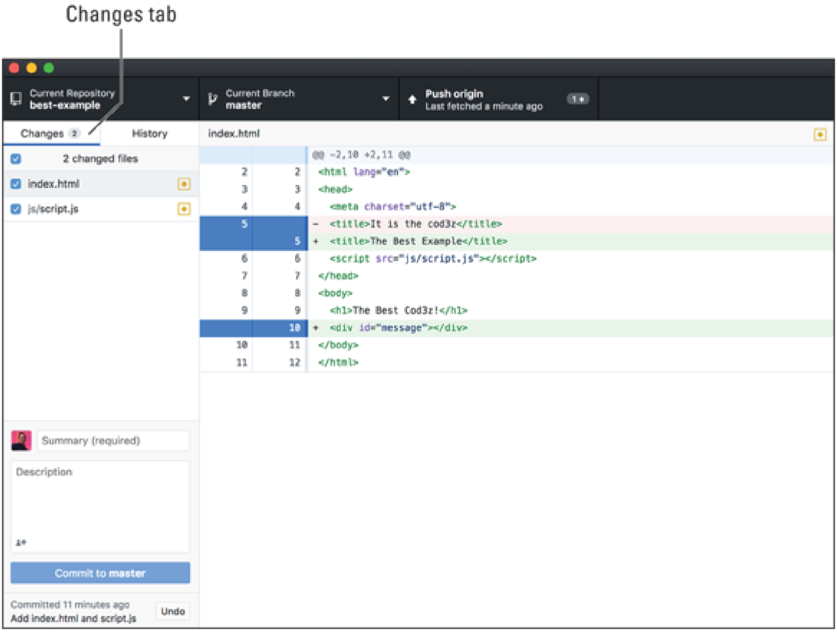
Make some changes so you have something to commit. In the example for this chapter, you can make some changes to `index.html` shown in bold.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>The Best Example</title>
  <script src="js/script.js"></script>
</head>
<body>
  <h1>The Best Cod3z!</h1>
  <div id="message"></div>
</body>
</html>
```

Update `script.js` to populate the new DIV element, like civilized people would, rather than pop up an alert message. Changes are in bold:

```
document.addEventListener(
  "DOMContentLoaded",
  function(event) {
    var message = document.getElementById('message')
    message.innerText = 'The script ran!'
  }
);
```

Switch back to Desktop and click the Changes tab, shown in Figure 7-5.


Changes tab
Current Repository
best-example
Current Branch
master
Push origin
Last fetched a minute ago
1
Changes 2
History
2 changed files
index.html
js/script.js
index.html
Summary (required)
Description
Commit to master
Undo

```
@@ -2,10 +2,11 @@
 <html lang="en">
 <head>
   <meta charset="utf-8">
-  <title>It is the cod3z</title>
+  <title>The Best Example</title>
   <script src="js/script.js"></script>
 </head>
 <body>
   <h1>The Best Cod3z!</h1>
+  <div id="message"></div>
 </body>
 </html>
```

Committed 11 minutes ago
Add index.html and script.js

FIGURE 7-5: The Changes View showing the uncommitted changes.

The left pane lists the set of changed files. If you select a file, you can see the specific changes to that file in the right pane, which is called the *diff view.*

You can commit all the changes as a single commit, but sometimes you will have unrelated changes. In this example, we have two unrelated changes:

- The change to the title in `index.html`
- The message changes to both `index.html` and `script.js`

Each of these changes should be in their own commit. How do we do that when `index.html` contains two unrelated changes? Fortunately, Desktop provides a nice way to commit a portion of the changes in a file. This process is known as a partial commit:

1. **Deselect all changes.**

   In the left pane, uncheck the check box next to the label `2 changed files` to deselect all changes.

2. **Select `index.html` in the left pane.**

   Click the filename in the left pane to display the changes for `index.html`.

3. **Select the title changes.**

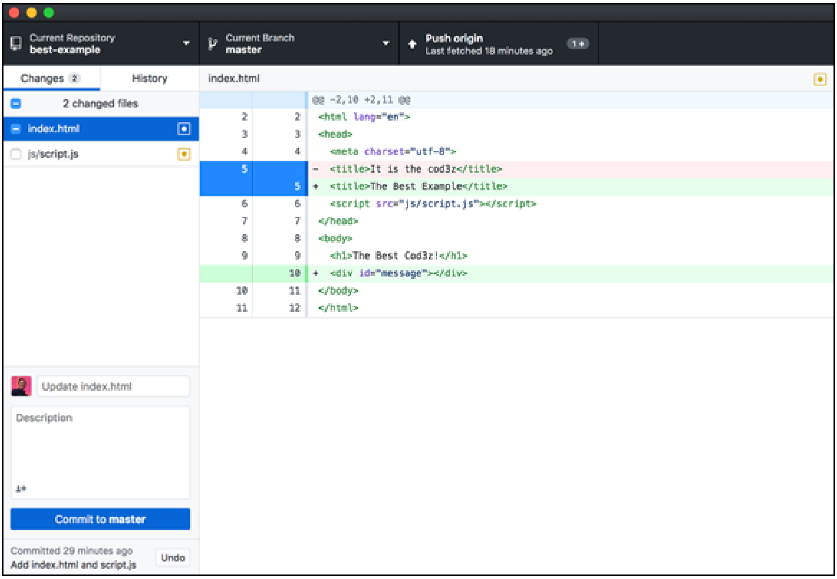
   In the diff view, click the line numbers in the gutter to select the changes you want to keep. To select a whole code block, click the thin line just to the right of the line number. Select the code block next to line 5 by clicking on the thin line next to line 5. After you select the code block, both lines labeled line 5 should be selected (selected lines show up as blue), as shown in Figure 7-6.

   WARNING You may be confused about why two lines are labeled 5 in the diff view. The numbers on the left represent what the file was originally named before you made the changes. The lines on the right represent the lines of the changed lines. Because we changed line 5, it's listed twice. Line 10 is a new line that didn't exist before, so it is listed only on the right.

4. **With those lines selected, enter a commit message and then click the Commit to master button.**

   As you can see in the bottom left portion of Figure 7-6, Desktop provides two fields for commit messages. Go ahead and enter “Change the title” into the summary and click the Commit to master button.

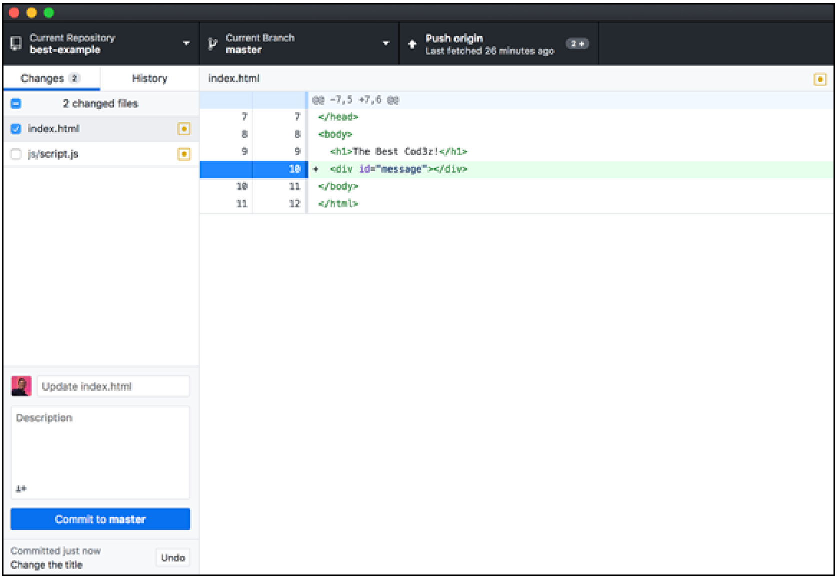
   Notice that the diff view updates to have the change only on line 10 (see Figure 7-7). That's because we committed the change on line 5.


Current Repository
best-example
Current Branch
master
Push origin
Last fetched 18 minutes ago
1↑
Changes 2
History
index.html
2 changed files
index.html
js/script.js
Update index.html
Description
Commit to master
Undo

```
@@ -2,10 +2,11 @@
 2   2   <html lang="en">
 3   3   <head>
 4   4     <meta charset="utf-8">
 5       -   <title>It is the cod3z</title>
     5   +   <title>The Best Example</title>
 6   6     <script src="js/script.js"></script>
 7   7   </head>
 8   8   <body>
 9   9     <h1>The Best Cod3z!</h1>
    10   +   <div id="message"></div>
10  11   </body>
11  12   </html>
```

Committed 29 minutes ago
Add index.html and script.js

FIGURE 7-6: The diff view with one change selected.


Current Repository
best-example
Current Branch
master
Push origin
Last fetched 26 minutes ago
2↑
Changes 2
History
index.html
2 changed files
index.html
js/script.js
Update index.html
Description
Commit to master
Undo

```
@@ -7,5 +7,6 @@
 7   7   </head>
 8   8   <body>
 9   9     <h1>The Best Cod3z!</h1>
    10   +   <div id="message"></div>
10  11   </body>
11  12   </html>
```

Committed just now
Change the title

FIGURE 7-7: The Changes tab after a partial commit.

If you’re following the example in this chapter, make sure all the remaining changes are selected by clicking the check box next to the label `2 changed files` until the check box is selected.

Using GitHub Conventions in Commit Messages

You can enhance your commit messages with GitHub-specific features, such as emojis, issue references, and coauthor credits.

Emojis

Emojis are little images or icons that convey an emotion or concept. Widely used on GitHub.com, emojis can bring a bit of levity and whimsy to an otherwise serious occupation.

In the commit summary box, you can initiate the emoji picker by typing the `:` character. If you keep typing, you can list all emojis that start with the letters you type. For example, Figure 7-8 lists all emojis that start with ar as the result of typing **:ar**.

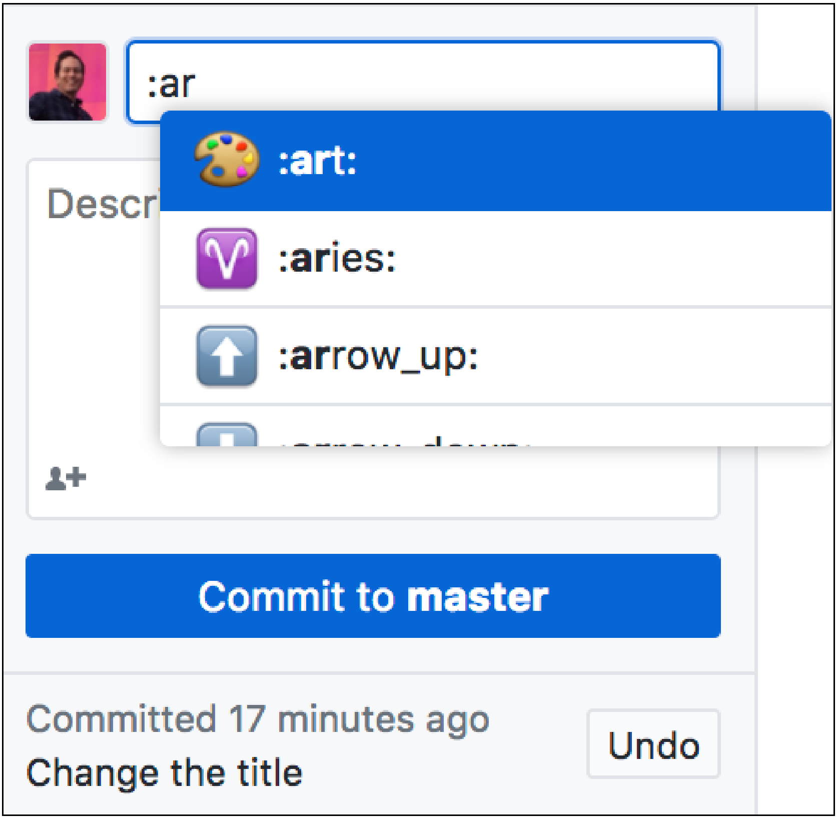

:ar
:art:
:aries:
:arrow_up:
Descr
Commit to master
Committed 17 minutes ago
Change the title
Undo

FIGURE 7-8: The emoji picker listing emojis.

You can select the one you want with the arrow keys and then press Tab to complete it. Desktop then fills in the full text of the emoji, which in this case is `:art:`.

Issue references

GitHub also lets you reference an issue in a commit message with the format #123 where 123 is the issue number. Desktop has support for looking up an issue when writing a commit message. To try this out, create an issue ahead of time so that you can reference the issue in a commit message. As an example, we created an issue that describes the need to replace the alert message with a better approach. We reference that issue in this commit message.

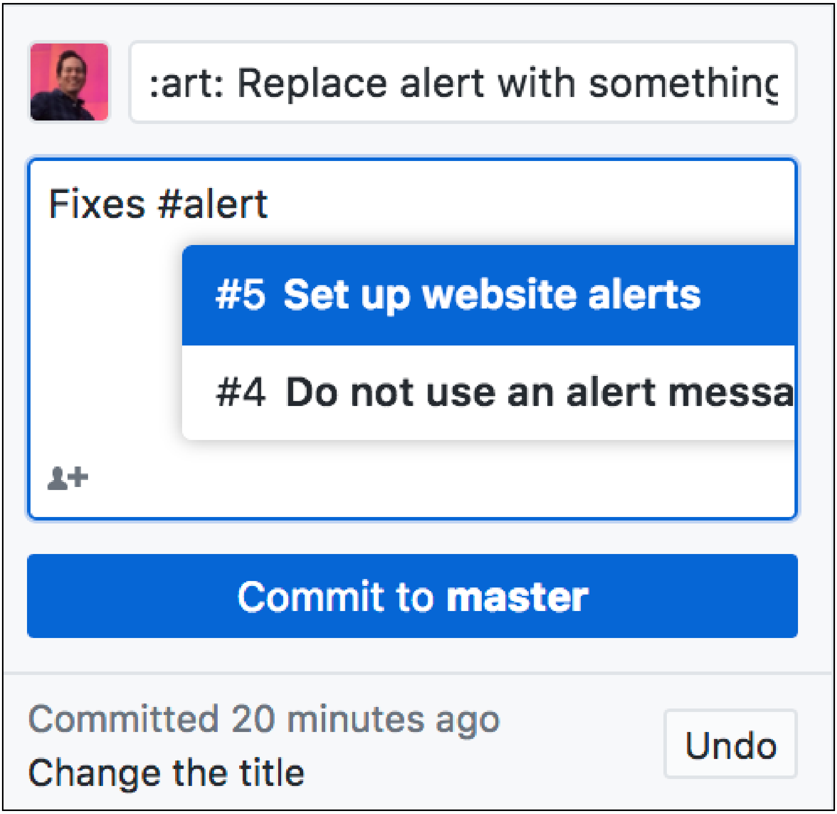
To reference an issue in a commit message:

1. **In the commit description field, type Fixes #.**

TIP A few recent issues appear. If you don't see the issue you want to reference and you don’t remember the issue number, You can start typing a word that’s in the issue that you remember. For example, when we type **#alert** a couple of issues pop up (see Figure 7-9).

2. **Select the issue you want to reference and press Tab.**

   In this example, we select issue 4. Desktop replaces `#alert` with `#4`.



:art: Replace alert with somethin

Fixes #alert

#5 Set up website alerts

#4 Do not use an alert messa

Committed 20 minutes ago

Change the title

FIGURE 7-9: A list of issues with the word “alert” in them.

Giving credit to coauthors

Git doesn't support multiple authors directly. However, the Git community created a convention for specifying multiple coauthors in a commit that is now supported by GitHub.

To give credit to coauthors:

1. **With the Desktop open, in the commit box with the Description label, click the little icon with a person and a plus sign in the bottom left corner.**

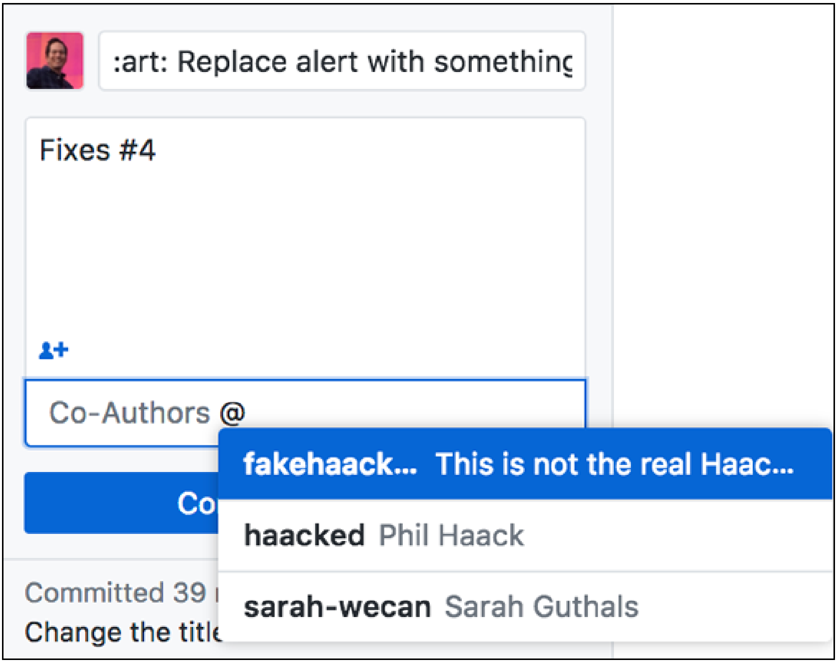
   Desktop adds a textbox to enter a coauthor’s GitHub username.

2. **Click the @ symbol to see a list of potential users, as shown in Figure 7-10.**

   GitHub lists only users who have access to the repository — for example, collaborators and org members (if the repository belongs to an organization).

   TIP Just like the issue selector, you can also search by first name, last name, or username by appending a bit after the @. For example, if I only remember y coauthor’s last name, I could type @guth to find my coauthor. Press `Tab`, and Desktop replaces whatever you typed so far with the selected user’s full username.

3. **To create the commit, click the Commit to master button.**



:art: Replace alert with something

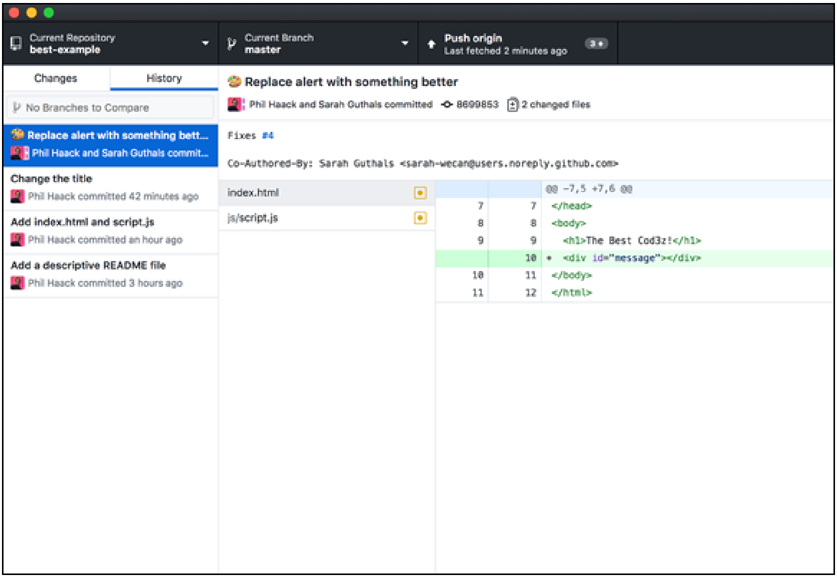
Fixes #4

Co-Authors @

fakehaack… This is not the real Haac…
haacked Phil Haack
sarah-wecan Sarah Guthals
Co
Committed 39
Change the title

FIGURE 7-10: A list of potential coauthors.

To see your commit, click the History tab and click the you just created (see Figure 7-11).


Current Repository
best-example
Current Branch
master
Push origin
Last fetched 2 minutes ago
3
Changes
History
No Branches to Compare
Replace alert with something bett...
Phil Haack and Sarah Guthals commit...
Change the title
Phil Haack committed 42 minutes ago
Add index.html and script.js
Phil Haack committed an hour ago
Add a descriptive README file
Phil Haack committed 3 hours ago
Phil Haack and Sarah Guthals committed
8699853
2 changed files
index.html
js/script.js

Replace alert with something better

```
Fixes #4

Co-Authored-By: Sarah Guthals <sarah-wecan@users.noreply.github.com>
```

```
          @@ -7,5 +7,6 @@
 7     7    </head>
 8     8    <body>
 9     9      <h1>The Best Cod3z!</h1>
      10  +   <div id="message"></div>
10    11    </body>
11    12    </html>
```

FIGURE 7-11: The newly created commit.

You can see in the commit message in the right pane that Sarah's username was replaced by the line

```
Co-Authored-By: Sarah Guthals <sarah-
   wecan@users.noreply.github.com>
```

REMEMBER That's the actual convention for specifying coauthors in Git commit messages. By using Desktop, you don't have to remember the exact format. You can just specify a username and let Desktop handle the rest.

Committing Code from Your Editor

Many editors have built-in support for committing code. Built-in support allows you commit code without having to switch to another application. The downside is that different editors have different levels of support for the various conventions you can use in a commit message.

But for quick and dirty commits, built-in support is very useful. Covering how every editor supports Git commits is out of the scope of this book, but you can see this in action with Atom in Chapter 5. For other editors, refer to their specific documentation.

FOR MORE READING

A lot of great guidance is out there for writing good commits. For example, the Git documentation at `https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project` has a section on contributing to a project and includes some Commit Guidelines.

I'm also a fan of a blog post by Chris Beams entitled "How to Write a Git Commit Message," which you can read at `https://chris.beams.io/posts/git-commit/`

Chapter 8

Working with Pull Requests

IN THIS CHAPTER

- **Creating a pull request**
- **Writing a great pull request**
- **Reviewing a pull request**
- **Exploring pull request workflows**

In Chapter 7, we note that a commit is the smallest unit of work with Git. On GitHub, the pull request is the main unit of work.

In this chapter, we explain exactly what a pull request is and how it pushes code to GitHub. We also describe the processes for opening, writing, and reviewing a pull request.

Understanding a Pull Request

The name pull request is confusing to some folks. “What exactly am I requesting to be pulled?” Good question. A *pull request* is a request to the maintainer of a repository to pull in some code.

When you write some code that you want to contribute to a repository, you create and submit a pull request. Your code contains some proposed changes to the target repository. A pull request is your way of offering these changes to the maintainer of the repository. It gives the repository maintainers an opportunity to review the changes and either accept them, reject them, or ask for more changes to be made.

Pushing Code to GitHub

To push code to GitHub, you need a repository. Open the repository of your choice. If you don’t have a repository yet, Chapter 7 walks through creating a repository that you can use.

TIP If you’d like to follow along with our example but you haven’t completed the steps in Chapter 7, fork the repository `https://github.com/FakeHaacked/best-example` and then clone your fork to your local machine. If that last instruction sounds like gobbledygook to you, you may want to review Chapter 6, which covers forking and cloning.

The first thing to do is create a new branch. Creating a new branch before writing new code is standard operating procedure with Git. I have a confession to make. We neglect to mention that in the example in Chapter 7. The example directs you to commit code directly to the `master` branch. That was a shortcut we suggested to keep things simple.

In this chapter, you'll do things the right way and work in a new branch. There’s an important reason for this. A pull request doesn’t consist of an arbitrary set of changes or commits. A pull request is always associated with a branch. In other words, a pull request is a request to merge one branch into another.

TIP While it’s true that a pull request can target any branch (except itself), the most common scenario is to target the main branch of a repository, typically named `master`.

This relationship between pull requests and branches is why you should create a new branch when starting new work. We name the branch `new-work` for this example, but feel free to name it whatever you want by replacing `new-work` with your own branch name in the following command:

```
$ git checkout -b new-work
```

Now that we have a branch, we need to create a commit in that branch. For this example, the specific contents of the commit are not important. You can choose any file and make some edits to the file, such as adding some text to the end. Or if you're following along with the repository we created in Chapter 7, manually edit the `README.md` file manually or run the following command to append some text to end of the file:

```
$ echo "\n## Installation" >> README.md
```

Now that you have a file with some changes, commit those changes. You can use the following command, for example, to commit all your changes. The commit message is not important here. The important thing is to have a commit in a branch to work with that is not in the `master` branch.

```
$ git add -A
$ git commit -m "Add text to the README"
```

Now push this new branch to GitHub.com (replace `new-work` with your branch name):

```
$ git push -u origin new-work
```

The `git push` command tells Git to push local commits to a remote repository. The `-u` flag specifies where to push it — in this case, to a branch also named `new-work` on the remote named `origin`.

The `-u` flag is only needed the first time you push a new branch to the server. From that point on, the `new-work` branch on your local machine is associated with the `new-work` branch on GitHub.com and any subsequent pushes to that branch do not need the flag.

Opening a Pull Request

Before you can open a pull request, your GitHub.com repository must have at least one branch other than the default branch. If you follow the steps in the earlier section “Pushing Code to GitHub,” you have a branch that is not yet merged into master. In our case, the branch is named `new-work` branch, but you may have named yours something else. Visit the repository on GitHub.com to open a pull request.

GIT ALIAS FOR OPENING GITHUB.COM FROM THE TERMINAL

Often when working with a repository in the terminal, you need to jump to the repository on GitHub.com. I have an alias just for this purpose. Chapter 6 covers aliases in more details and how to add them. You can run the following in the terminal to add a new `git browse` alias.

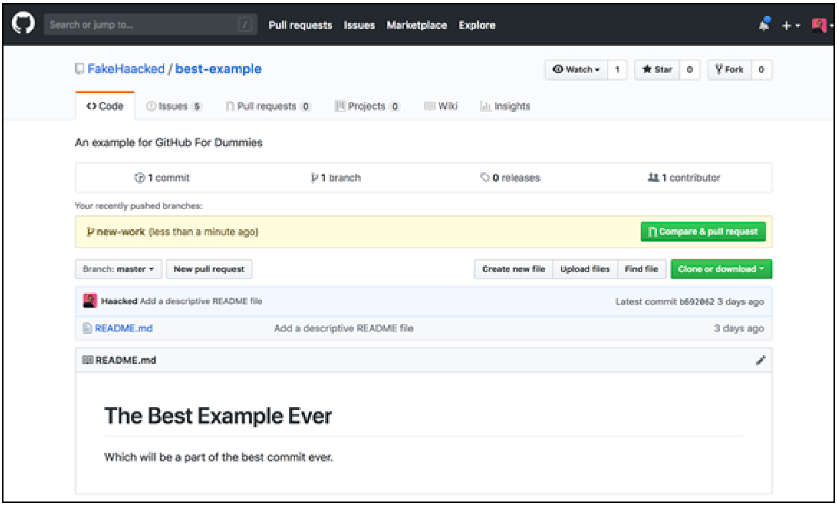
```
$ git config --global alias.browse '!open `git config
   remote.origin.url`'
```

To use the alias, just run the following in the terminal from within your repository directory.

```
$ git browse
```

This code launches your default browser and navigates to your repository's origin URL. This alias assumes you're using https for your git remotes and not SSH.

When you visit the repository on GitHub.com, you see a new message in the section labeled Your recently pushed branches, as shown in Figure 8-1.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Watch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 0
Projects 0
Wiki
Insights
1 commit
1 branch
0 releases
1 contributor
Your recently pushed branches:
new-work (less than a minute ago)
Compare & pull request
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
README.md

An example for GitHub For Dummies

Haacked Add a descriptive README file — Latest commit b692862 3 days ago

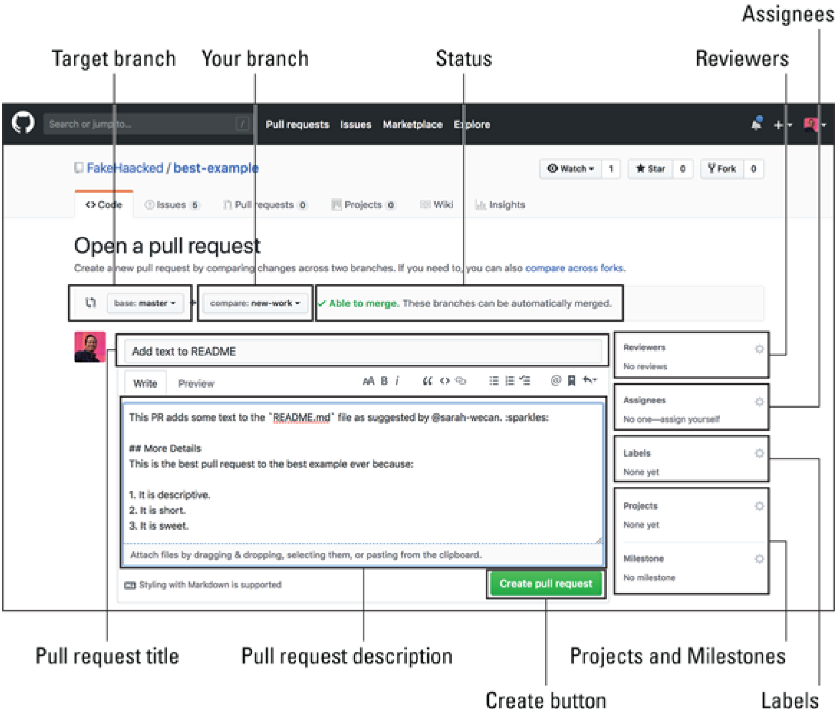
README.md — Add a descriptive README file — 3 days ago

The Best Example Ever

Which will be a part of the best commit ever.

FIGURE 8-1: Repository home page with recently pushed branches listed.

Click the Compare & pull request button to navigate to the Open a pull request page, as shown in Figure 8-2. The target branch (the branch you want to pull your changes into) is the default branch for the repo. Your branch is listed next to the target branch, and a status of whether your branch can be merged into the target branch is next to that. The pull request title is the same as the most recent commit message — in this example Add text to the README — and your description is blank. Figure 8-2 shows an example description.


Assignees
Target branch
Your branch
Status
Reviewers
Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Watch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 0
Projects 0
Wiki
Insights
Write
Preview
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Styling with Markdown is supported
Create pull request
Pull request title
Pull request description
Projects and Milestones
Create button
Labels

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ← compare: new-work ✓ Able to merge. These branches can be automatically merged.

Add text to README

```
This PR adds some text to the `README.md` file as suggested by @sarah-wecan. :sparkles:

## More Details
This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.
```

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

FIGURE 8-2: The Open a pull request page.

TIP You can change the default branch by choosing Settings ⇒ Branches section of your repository.

GIT ALIAS FOR OPENING THE BRANCH COMPARE WEB PAGE

If, for some reason, the branch for which you want to create a pull request isn’t listed in the recently pushed branches section, you can navigate to the Compare page for your branch, which has the format `https://gitub.com/{owner}/{repo}/compare/{BRANCH}`. As luck would have it, I have an alias for that:

```
$ git config --global alias.pr '!f(){ URL=$(git config
    remote.origin.url); open ${URL%.git}/compare/$(git
    rev-parse --abbrev-ref HEAD); }; f'
$ git pr
```

Just as with the git browse alias earlier in the sidebar “Git Alias for opening GitHub.com from the terminal”, this alias assumes you use HTTPS URLs for your remotes and not SSH.

Describing the pull request

From the Open a pull request page, you can enter a summary and description. Chapter 7 covers some GitHub conventions for commit messages. Most of those conventions are also supported in a pull request — for example, mentioning people using the `@USERNAME` format. In Figure 8-2, I mention `@sarah-wecan`. When I create the pull request, `@sarah-wecan` receives a notification.

You can reference issues and other pull requests using the `#ISSUEID` format. And, of course, you can add emojis, such as `:sparkles:`.

In this section, we cover what to write here in more detail. There's a lot that goes into a good pull request.

Adding reviewers

To the right of the pull request summary and description fields is a set of options for the pull request.

The first one, Reviewers, lets you specify one or more people that you want to review your pull request. To add reviewers:

1. **Click the gear to see a list of people you can mention.**

   For repositories with a large number of users, you can start typing to filter the set of users.

2. **Click each user to add them to the list of reviewers.**

When you add a reviewer, they're immediately notified when you finish creating the pull request. Figure 8-3 shows the reviewers list with two reviewers selected.

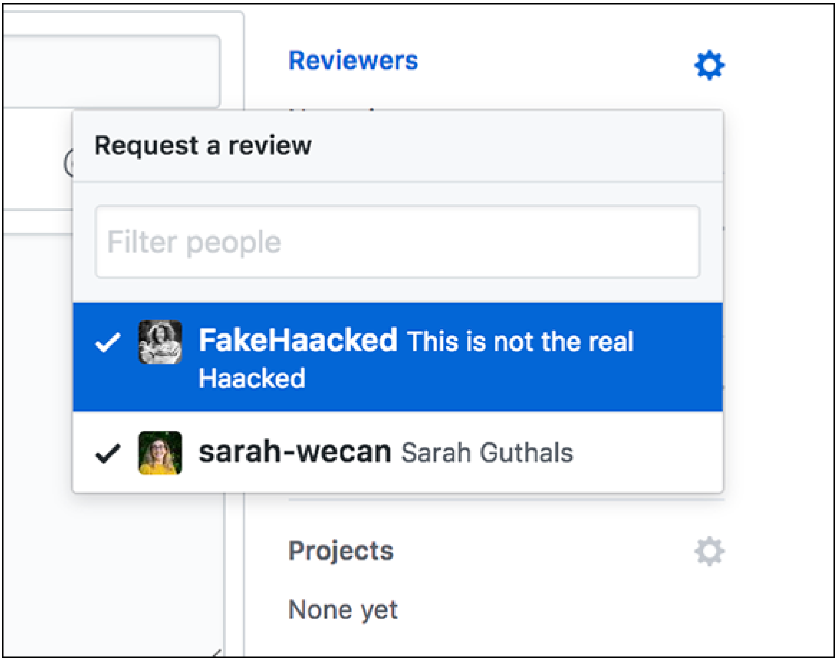

Reviewers
Request a review
Filter people
✓ FakeHaacked This is not the real Haacked
✓ sarah-wecan Sarah Guthals
Projects
None yet

FIGURE 8-3: List of potential reviewers with two reviewers selected.

Specifying assignees

The next option after Reviewers is an option to specify assignees. An *assignee* is the person who should take action on the pull request. Often, a pull request represents a work in progress and not the final result of some work. If more work needs to be done on a pull request, you'd assign the pull request to the person that should do the work.

To specify assignees:

1. **Click the gear to see a list of assignees.**

   The assignee dialog box works just like the reviewers dialog box, described in the preceding section. It allows you to select one or more assignees.

2. **Click each user to add them to the list of reviewers.**

TIP In most cases, it's best to just assign one person who will be responsible for the next step. Assigning one person reduces the chances that multiple assignees all think the other assignees are responsible for the work.

Specifying labels

Chapter 3 introduces labels on issues as a way to figure out what to work on next. Labels work the same way for pull requests. They provide convenient grouping and context to help you decide what to work on next or what to review next.

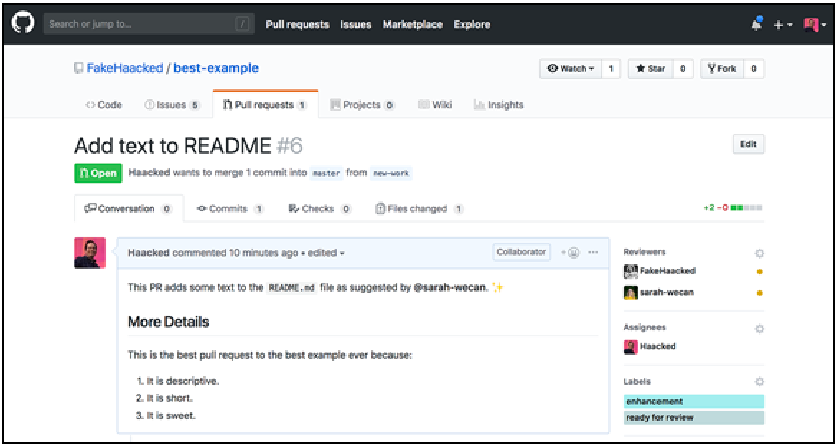
The set of labels you can use on issues and pull requests are the same, but some labels make more sense for issues than pull requests and vice versa. For example, many repositories have a "ready for review" label specifically for pull requests.

Specifying projects and milestones

The last two options allow you to specify the project board and milestone that this pull request belongs to. Chapter 3 covers projects, and Chapter 11 covers milestones.

Creating the pull request

After you have the pull request written and specified all the pull request options, click the Create pull request button to save all your work and create the pull request. Figure 8-4 shows a pull request I created on my repository.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Watch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Wiki
Insights
Edit
Conversation 0
Commits 1
Checks 0
Files changed 1
+2 −0
Collaborator
Reviewers
FakeHaacked
sarah-wecan
Assignees
Haacked
Labels
enhancement
ready for review

Add text to README #6

Open Haacked wants to merge 1 commit into `master` from `new-work`

Haacked commented 10 minutes ago • edited

This PR adds some text to the `README.md` file as suggested by @sarah-wecan. ✨

More Details

This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.

FIGURE 8-4: A created pull request.

Writing a Great Pull Request

Writing a great pull request is a bit of an art. For an open source project, much of the project's communication with people occurs within pull requests.

If you're contributing to a project, your pull request is your chance to make a strong case for why your code should be pulled into the main branch. Make sure to put your best foot forward.

Knowing your audience

Before you write a single word, understanding your audience is helpful so that you can focus your words on what is most useful. A pull request may serve many audiences. Keeping all of your audiences in mind is important, but your primary focus is on the folks who will review and make a decision on whether your pull request will be merged. You want to make their lives easier as they tend to be very busy.

Even though the project maintainers are your primary audience, you should never forget that many others will read the pull request. For an open source project, that audience may be the entire world. So keep your language respectful, friendly, and inclusive.

WARNING It's pretty common to have someone write a pull request in a fit of anger and later regret the words they use. So if you happen to be rage coding, take a moment to cool down and gather your thoughts before creating the pull request.

Making the purpose clear

Make sure to be concise and informative. For example, the summary should make the purpose of the pull request clear. The summary is the only part shown on the page that lists pull requests. It needs to be scannable.

Here are some examples of good pull request summaries:

- Adds the About page to the website
- Minimize boilerplate setup code for JavaScript libraries
- Extract and isolate error handling from GitStore internals

Here are some bad examples taken from my own repositories.

- Teams are forever
- Typo
- Small changes

TIP The description should provide a bit more explanation about the purpose of the pull request. Don't write a book, but do make it clear what the pull request attempts to accomplish.

Keeping it focused

Much like a commit, a pull request should not contain a bunch of unrelated changes. A pull request may consist of multiple commits, but they should all be related to the task at hand.

You can often tell that a pull request is doing too much when writing a concise description of what the pull request accomplishes is difficult.

Even if the pull request is focused on a single major change, keep the pull request to a manageable size. Reviewing a very large pull request is difficult.

If the pull request addresses a very large task, break down the task into smaller steps and submit pull requests for each step.

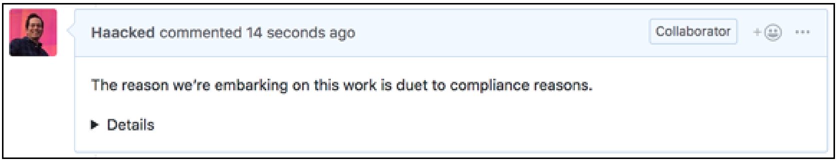
Explaining the why

The previous suggestion, "Keeping it focused," focuses on what the pull request does. You also need to explain why you're taking on this work. The pull request description is an opportunity to provide links to other documents that provide more context. You can't assume everyone will be familiar with the history.

If you have a lot of context to share, you can provide a brief summary followed by more details within a `<details>` tag. For example, if you add a pull request comment with the following text:

```
The reason we're embarking on this work is due to compliance
    reasons.
<details>
## More Details
I don't want to bore everyone with all the nitty gritty
    details, but for those who are interested, keep on reading…
</details>
```

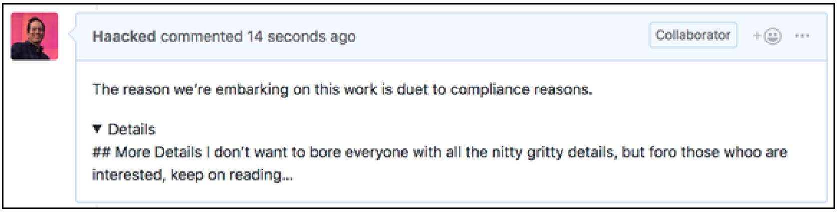
GitHub displays the details section collapsed by default, as shown in Figure 8-5.



The reason we're embarking on this work is duet to compliance reasons.

FIGURE 8-5: Details section collapsed.

Click the details label to expand the details section, as shown in Figure 8-6.


Haacked commented 14 seconds ago
Collaborator

The reason we're embarking on this work is duet to compliance reasons.

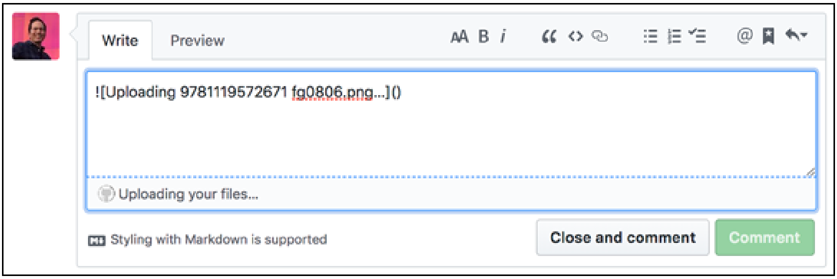
▼ Details

More Details I don't want to bore everyone with all the nitty gritty details, but foro those whoo are interested, keep on reading…

FIGURE 8-6: Details section expanded.

A picture is worth a thousand words

GitHub supports adding images to a pull request description by dragging and dropping an image. When you drag and then drop the image on the text field, GitHub uploads the image and replaces it with the markdown for rendering an image. Figure 8-7 shows an upload in progress after I dragged an image into a pull request comment field.


Write
Preview
AA B i

```
![Uploading 9781119572671 fg0806.png…]()
```

Uploading your files…
Styling with Markdown is supported
Close and comment
Comment

FIGURE 8-7: Uploading an image to a pull request.

Visit `https://github.com/FakeHaacked/best-example/pull/6#issuecomment-454927796` to see this comment. It's very meta as it's a screenshot of the same repository it's a comment on.

TIP If an image is worth a thousand words, an animated gif is worth even more. If you can create an animated gif that demonstrates the behavior change introduced by the pull request, adding that gif to the pull request is usually very appreciated by those who review it.

Including a call to action

You need to be very clear about what feedback you want from others on the pull request. For example, if the pull request is a work in progress, make that clear from the start so that people don't waste time reviewing a pull request that isn't ready for review.

To make that clear, follow the conventions of the repository. You can find out the conventions by orienting yourself with the repository as described in Chapter 5.

Following the conventions is important so that others know what is expected of them with respect to your pull request.

WRITING A PERFECT PULL REQUEST

The previous suggestions describe writing a great pull request. It's not comprehensive. For more suggestions, check out a blog post on the GitHub blog entitled "How to write the perfect pull request" at `https://blog.github.com/2015-01-21-how-to-write-the-perfect-pull-request/`.

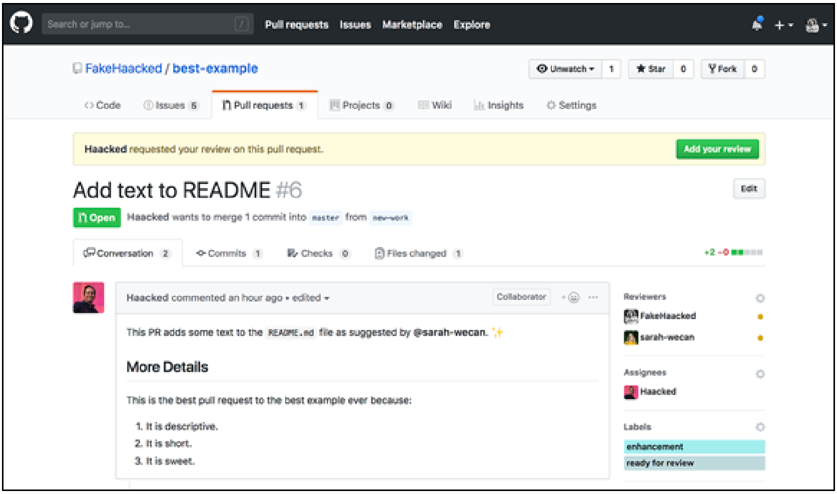
Reviewing a Pull Request

Pull requests have two parties involved: the person who writes and opens the pull request, and the person (or people) who reviews it. In this section, you put yourself in the position of a reviewer of a pull request.

REMEMBER Reviewing a pull request is a very active activity. It takes a lot of focus and attention to do it well. It doesn't serve the folks submitting pull requests if you just take a cursory look at it, comment LGTM (Looks Good To Me), and don't provide quality feedback.

Sometimes, a cursory look is all you have time for. In that case, make it clear what you did and didn't review and suggest that someone else provide a more detailed review.

When someone adds you as a reviewer, you'll receive a notification about the request, typically via email or on your notification bell at the top right of GitHub.com. If you visit the pull request, you'll see a banner message, as shown in Figure 8-8.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Wiki
Insights
Settings
Add your review
Edit
Open
Conversation 2
Commits 1
Checks 0
Files changed 1
+2 −0
Collaborator
Reviewers
FakeHaacked
sarah-wecan
Assignees
Haacked
Labels
enhancement
ready for review

Haacked requested your review on this pull request.

Add text to README #6

Haacked wants to merge 1 commit into `master` from `new-work`

Haacked commented an hour ago • edited

This PR adds some text to the `README.md` file as suggested by @sarah-wecan. ✨

More Details

This is the best pull request to the best example ever because:

1. It is descriptive.
2. It is short.
3. It is sweet.

FIGURE 8-8: Message requesting your review on this pull request.

WARNING Don't click the Add your review button just yet. That takes you to a dialog box to write a write-up of your review. Only do that when you're ready to complete your review.

Reviewing the Conversation tab

When you review a pull request, the first thing to do is read through the contents of the Conversation tab for the pull request. Make sure that you understand the purpose of the pull request and why it's necessary.

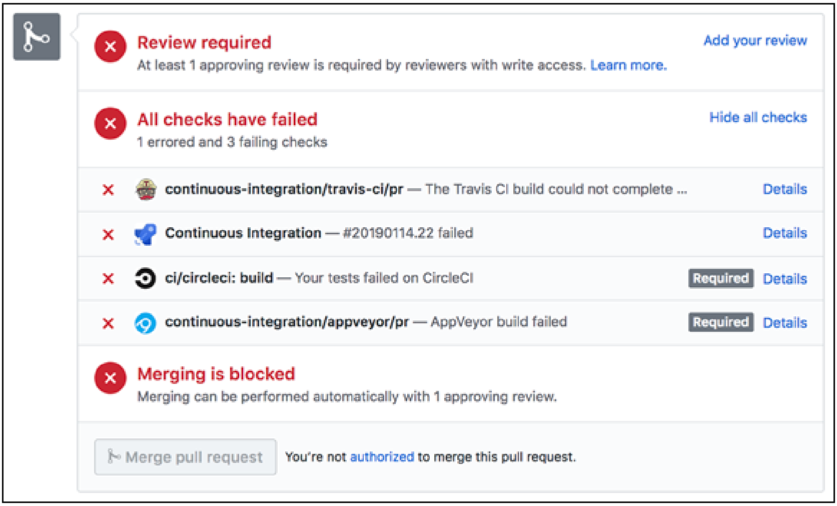
At the bottom of the conversation is a section that displays the checks that GitHub runs against the repository if there are any. If no checks are set up, you see the message `Continuous integration has not been set up`. The status of these checks is the next thing you should check. Many repositories will have several checks, such as does the code compile, did all the tests pass, and so on.

If any of these tests fail, you shouldn't spend any more time reviewing the code. The person submitting the pull request can see that their pull request doesn't pass all the checks, and it's up to them to fix it.

TIP

Even though the person submitting a pull request should be able to see that the checks have failed, sometimes they don't stick around long enough after creating the pull request to see that. You may want to add a gentle note that mentions the person submitting the pull request and informs them that the checks have failed and he should try fixing the problems and push his changes to the pull request branch again.

Figure 8-9 shows an example from the GitHub Desktop open source project of a pull request that fails the continuous integration (CI) builds.



Review required
At least 1 approving review is required by reviewers with write access. Learn more.
Add your review

All checks have failed
1 errored and 3 failing checks
Hide all checks

✗ continuous-integration/travis-ci/pr — The Travis CI build could not complete ... Details

✗ Continuous Integration — #20190114.22 failed Details

✗ ci/circleci: build — Your tests failed on CircleCI Required Details

✗ continuous-integration/appveyor/pr — AppVeyor build failed Required Details

Merging is blocked
Merging can be performed automatically with 1 approving review.

Merge pull request You're not authorized to merge this pull request.

FIGURE 8-9: Pull request with failed checks.

Reviewing the changed files

Assuming you're happy with the contents of the Conversation tab and all the checks pass, it's time to get into nitty gritty and review all the file changes. Click the Files changed tab to see all the changes in the pull request.

At this point, you should review the code for things like

- **Clarity:** Is the code easy to read and understand? Could it be made more clear? there appropriate comments throughout the code? Obtuse, hard-to-understand code becomes a maintenance nightmare down the road.
- **Correctness:** Does the code do what the pull request says it does? Are there any glaring bugs? Are there any errors of omission? Are there any tests missing?
- **Security:** Related to the previous item, a security review requires a specific mindset. Ideally, you work with security experts who can help review the code for security. The idea here is to think about all the ways a bad actor could attempt to attack the code. There are many frameworks for doing security review, such as STRIDE. You should also think about how bad actors can use the code to harm other users. Does the code protect users privacy? Does it ask for consent to take actions on behalf of users.
- **Conventions and idioms:** Just because code is correct, it doesn't mean it's necessarily idiomatic. A code review is a good place to teach and learn conventions and idioms specific to a project.

In the last section, we mention conventions. By conventions, we mean common approaches to accomplishing a task. For example, if your project has a certain approach to querying the database, make sure the code follows that approach.

WARNING One thing to note is we don't cover code style issues in the list of things to review. A code review should not cover nitpicks like whether a curly brace goes on its own line or not. An overly pedantic nitpicky code review does not set a friendly and collaborative tone. Depending on the context, you can just fix these things yourself or better yet, use automated tools, such as a linter or prettifier, to do it. You'll save everyone a lot of time and headaches.

Commenting on code

When reviewing changed files, you can add comments to specific lines of code to indicate a problem with the code, add a suggestion to make it better, or celebrate someone's awesome code writing skills with a

`:sparkle:`.

TIP Positive and encouraging comments set for a welcoming and collaborative tones. Maintainers often forget how daunting it can be to contribute code to a project for the first time. Don’t be stingy with the `:sparkle:` emojis!

To comment on code:

1. **Hover your mouse over the line of code.**

   A blue square with a plus sign appears next to the line number.

2. **Click the square to reveal a comment form for that specific line of code, as shown in Figure 8-10.**

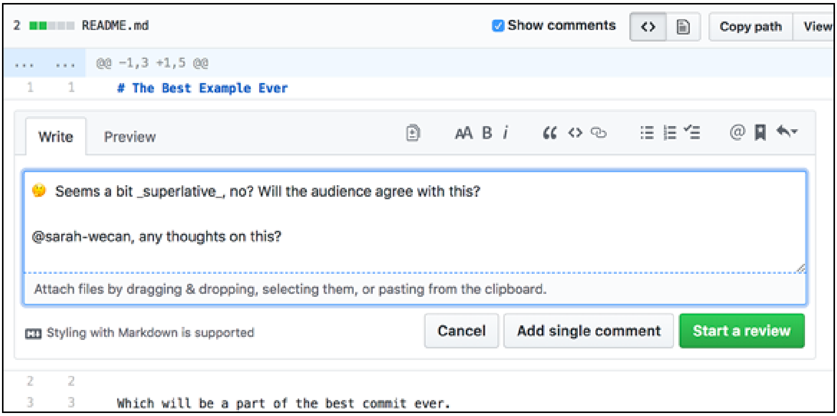
   The comment form supports the same things issues and pull requests do, such as mentions, Markdown, and, of course, emoji.

   At this point, you can choose to click Add single comment or Start a review.

   Clicking Add single comment immediately adds your comment to the pull request and sends out any notifications. This can be useful when making a quick one-off comment. In most cases, we recommend against this approach as it doesn't lead to well considered and thoughtful code reviews.

3. **Click Start a review instead to create a review and add the comment as a pending comment to the review, as shown in Figure 8-11.**

   The pending label indicates that you’re the only one who can see the comment so far. By starting a review in this manner, you can continue to add (and edit) pending comments as you review the code and only publish them when you’re completely satisfied with the review.



```
@@ -1,3 +1,5 @@
# The Best Example Ever

Which will be a part of the best commit ever.
```

Seems a bit _superlative_, no? Will the audience agree with this?

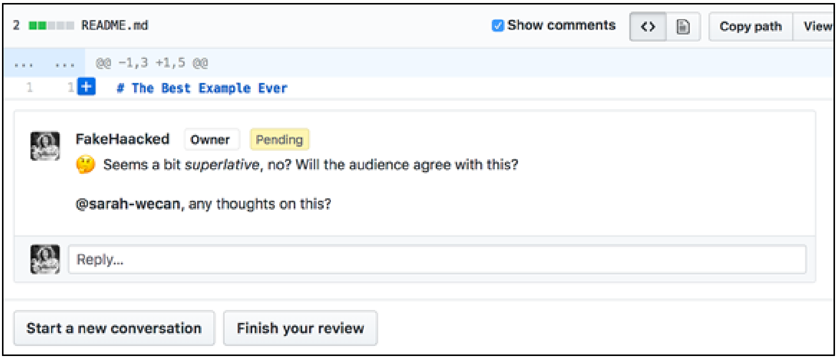
@sarah-wecan, any thoughts on this?

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

2
README.md
☑ Show comments
Copy path
View
Write
Preview
Cancel
Add single comment
Start a review

FIGURE 8-10: Pull Request Comment form for a line of code.



```
@@ -1,3 +1,5 @@
# The Best Example Ever
```

FakeHaacked Owner Pending

Seems a bit *superlative*, no? Will the audience agree with this?

@sarah-wecan, any thoughts on this?

2
README.md
☑ Show comments
Copy path
View
Reply...
Start a new conversation
Finish your review

FIGURE 8-11: Pending comment in a pull request review.

TIP

The reason we recommend starting a review as opposed to adding single comments as you go is a review leads to a more coherent and helpful code review. A common occurrence when reviewing code is that something you read later in the review makes you realize a comment you made earlier should be updated or even deleted. A review lets you make those adjustments before you send anything to the author. This option gives you an opportunity to review your review before publishing it.

Suggesting changes

Sometimes as you review code, you come across a section of code where you feel like it'd be faster to just fix the code than try to explain in words what should be fixed.

Or perhaps you run into a lot of small errors, such as typos, where commenting on each one produces more noise than just fixing them. For these situations, GitHub supports suggesting a change via a pull request comment that the author can apply with a click.

To suggest a change:

1. **Opening the comment form for the line of code you want to change.**

   When you bring up the comment form, you see an icon with a + and - symbol. When you hover over the symbol, the tooltip describes the purpose of this button (see Figure 8-12). The tooltip also describes a keyboard shortcut you can use to suggest a change, `CMD + G`.

2. **Click the Suggest changes button.**

   GitHub adds a suggestion block into the body of your comment with the current contents of the line you want to change.

3. **Change the contents of the block to suggest what the final result should be.**

   Figure 8-13 shows an example where I suggest a change to add the word "Instructions" to the line.

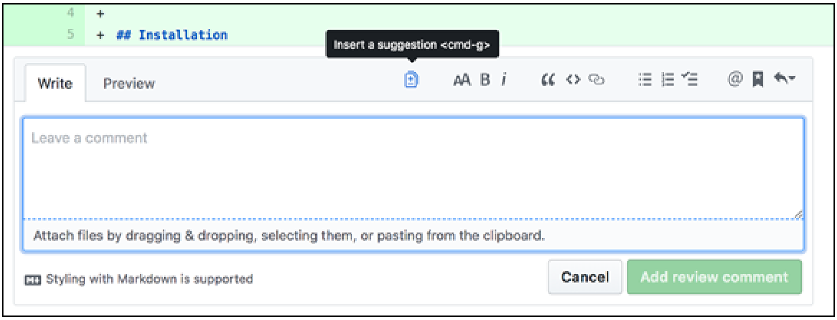
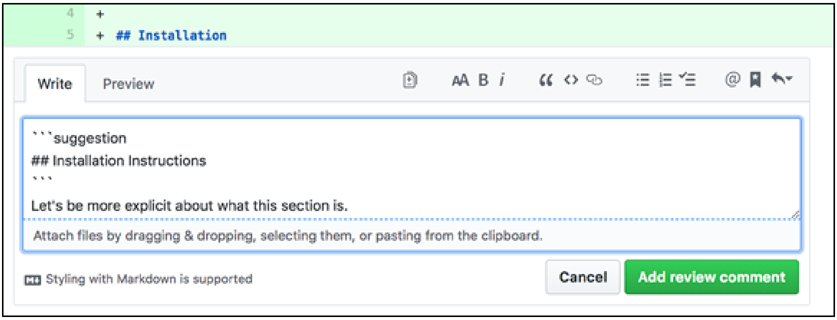

```
4  +
5  + ## Installation
```
Insert a suggestion <cmd-g>
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Cancel
Add review comment

FIGURE 8-12: Comment form with the Suggest changes button.


```
4  +
5  + ## Installation
```
Write
Preview

~~~
```suggestion
## Installation Instructions
```
Let's be more explicit about what this section is.
~~~

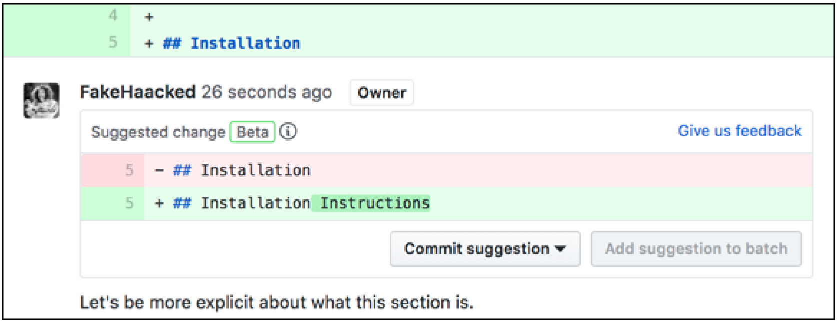
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Cancel
Add review comment

FIGURE 8-13: Comment with a suggested change.

TIP

Anything you write outside of the suggestion block in the comment is not included in the suggested change. This lets you provide a reason for the suggested change. Except for truly simple or self-explanatory suggestions, we recommend always providing this extra context.

After you create a comment with a suggested change, GitHub displays the change as a small diff within the comment. If the person viewing the suggestion has commit access, they will see an option to commit the suggested change (see Figure 8-14). They will also thank you for saving them time!



```
+
+ ## Installation
```

FakeHaacked 26 seconds ago

```
- ## Installation
+ ## Installation Instructions
```

Let's be more explicit about what this section is.

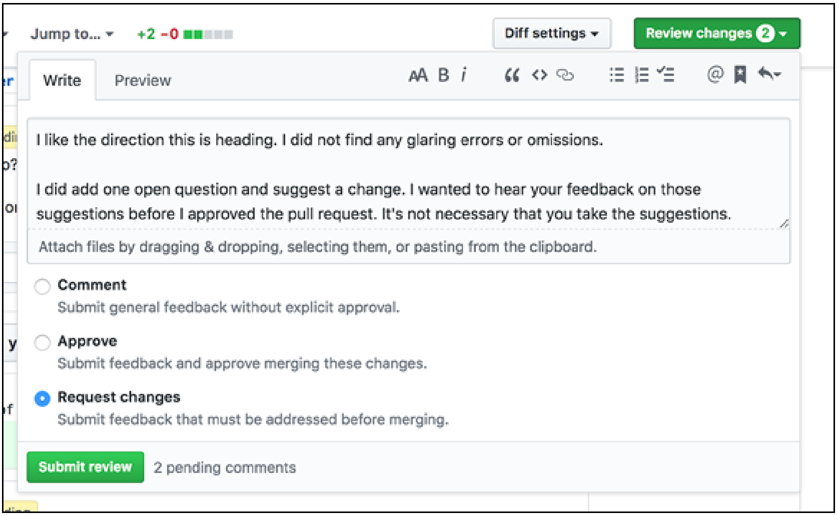
FIGURE 8-14: Comment with a suggested change.

Finishing the review

After you have made all your suggestions, you can finish the review so the author receives all your valuable feedback and can start to address it. To finalize the review, click the Finish your review button at the bottom of any pending comment (refer to Figure 8-11).

You see a form where you can write your overall summary about the pull request. This form is an opportunity to bring up any review comments that are not specific to any lines of code. It is also a good opportunity to offer some general praise, raise broad concerns, suggest follow-up actions, and so on.

After you type your comment, check one of the options to indicate your position on the pull request. Figure 8-15 shows the comment form with the review options.



I like the direction this is heading. I did not find any glaring errors or omissions.

I did add one open question and suggest a change. I wanted to hear your feedback on those suggestions before I approved the pull request. It's not necessary that you take the suggestions.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

○ **Comment**
Submit general feedback without explicit approval.

○ **Approve**
Submit feedback and approve merging these changes.

● **Request changes**
Submit feedback that must be addressed before merging.

FIGURE 8-15: Comment with a suggested change.

TIP If you're reviewing your own pull request, the only option available is to Comment on the pull request.

Your choice may determine whether this pull request can be merged into the default branch. Whether your choice blocks a merge depends on how the branch protection rules in place for the repository. For example, some repositories may require a certain number of approved reviews before the pull request can be merged. Go to Settings ⇒ Branches to manage branch protection rules.

Reading More About Pull Requests

Pull requests and code reviews are a very important part of the software development lifecycle. As such, a lot of great advice is out there for doing these things well — more advice than we can offer in a single chapter. Here's some more articles to read to help you get the most out of pull requests and code review.

- **Building an Inclusive Code Review Culture:** How you review code is a reflection of your overall culture. This article describes an approach that is inclusive and collaborative.

`https://blog.plaid.com/building-an-inclusive-code-review-culture/`

- **Code Review Like You Mean It:** This article is a brief write-up about the efficacy of code review along with some tips on how to do them well.

  `https://haacked.com/archive/2013/10/28/code-review-like-you-mean-it.aspx/`

Part 4

Manage and Contribute to Large Projects

IN THIS PART …

Find open source software projects (OSS) on GitHub.com.

Discover effective strategies for contributing to OSS.

Read and understand contributing guidelines for large projects.

Set up a repository so that it’s ready for an OSS project.

Discover strategies from OSS that are useful for large, private projects.

Create a team on GitHub to better manage your inner-source project.

Chapter 9

Exploring and Contributing to OSS

IN THIS CHAPTER

- **Exploring OSS projects on GitHub**
- **Contributing to OSS**
- **Contributing guidelines**

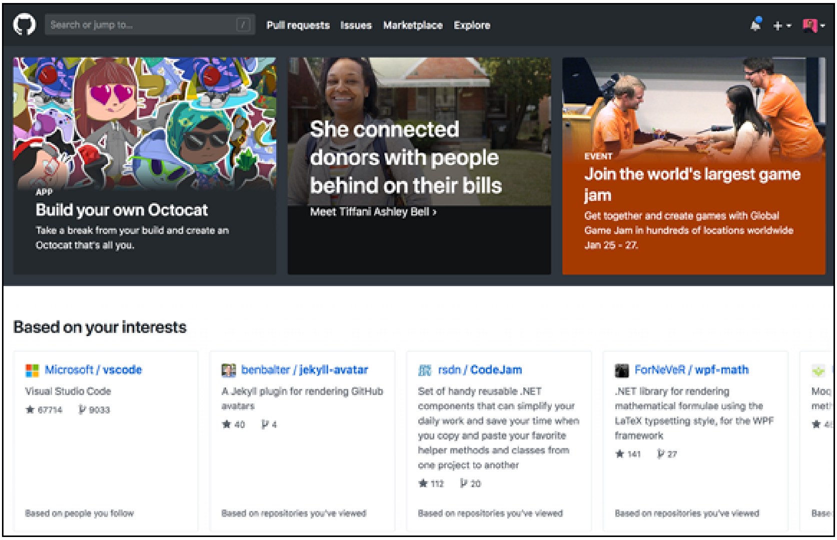
For developers, GitHub is the greatest source of treasure ever created, if you know where to look. An open source repository exists for every possible need out there. This cornucopia of choice can be overwhelming at first.

Going beyond just making use of open source software (OSS), contributing to open source is a fantastic way to continue your development as a software developer. It gives you the opportunity to work with technologies that you may not otherwise work with in your day job. It connects you with a large community of developers working on a diverse array of challenges. Many of these developers are happy to share their knowledge with folks looking to learn something new.

In this chapter, we look at ways to explore the range of open source software GitHub has to offer. We not only look at ways of discovering repositories you may want to use in your own projects, but also provide tips for finding repositories you may want to contribute to.

Exploring GitHub

The Explore page on GitHub, located at `https://github.com/explore`, is a great starting point to discover repositories that may align with your interests. It contains a mix of human and algorithmically curated content. Figure 9-1 shows a portion of the Explore page.



APP

Build your own Octocat

Take a break from your build and create an Octocat that's all you.

She connected donors with people behind on their bills

Meet Tiffani Ashley Bell ›

EVENT

Join the world's largest game jam

Get together and create games with Global Game Jam in hundreds of locations worldwide Jan 25 - 27.

Based on your interests

Microsoft / vscode
Visual Studio Code
★ 67714 ⑂ 9033
Based on people you follow

benbalter / jekyll-avatar
A Jekyll plugin for rendering GitHub avatars
★ 40 ⑂ 4
Based on repositories you've viewed

rsdn / CodeJam
Set of handy reusable .NET components that can simplify your daily work and save your time when you copy and paste your favorite helper methods and classes from one project to another
★ 112 ⑂ 20
Based on repositories you've viewed

ForNeVeR / wpf-math
.NET library for rendering mathematical formulae using the LaTeX typesetting style, for the WPF framework
★ 141 ⑂ 27
Based on repositories you've viewed

FIGURE 9-1: The Explore page on GitHub.

In the next several sections, we describe each section of the Explore page and its significance. Where it makes sense, we cover what you need to do to make your repository a candidate for inclusion in the section.

Exploring the headline section

The headline section contain three featured sites. At the time of writing, it featured

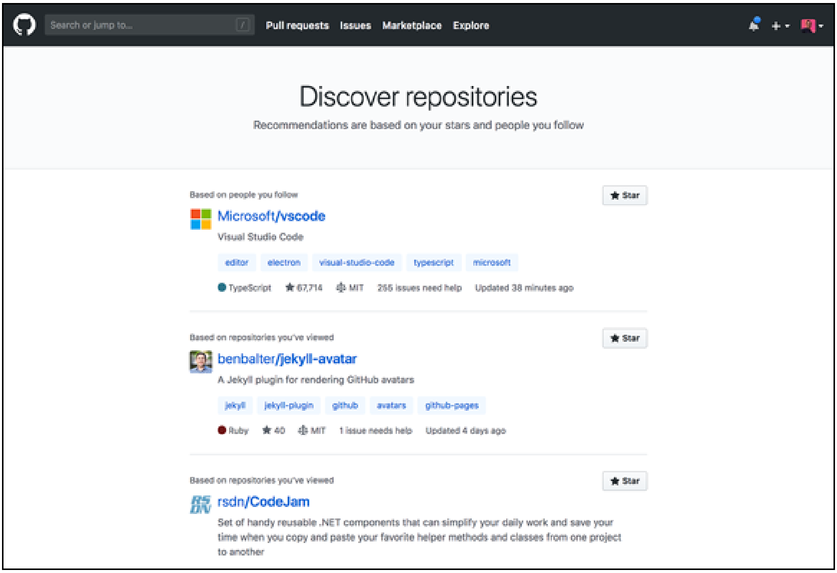
- A link to Build Your Own Octocat app at `https://myoctocat.com`
- A profile of Tiffany Ashley Bell, founder of the Detroit Water Project
- A link with information about an event, The Global Game Jam

You can check back on this section to discover interesting apps, people, and events that GitHub feels are worth featuring.

Discovering repositories

The next section under the headlines displays a selection of repositories based on your interests. This selection is pulled from the Discover page (`https://github.com/discover`), an algorithmically curated page of recommendations specific to you.

Click the heading, based on your interests, to see the full list of your recommendations (see Figure 9-2).



Discover repositories

Recommendations are based on your stars and people you follow

Based on people you follow

Microsoft/vscode

Visual Studio Code

editor electron visual-studio-code typescript microsoft

TypeScript ★ 67,714 MIT 255 issues need help Updated 38 minutes ago

Based on repositories you've viewed

benbalter/jekyll-avatar

A Jekyll plugin for rendering GitHub avatars

jekyll jekyll-plugin github avatars github-pages

Ruby ★ 40 MIT 1 issue needs help Updated 4 days ago

Based on repositories you've viewed

rsdn/CodeJam

Set of handy reusable .NET components that can simplify your daily work and save your time when you copy and paste your favorite helper methods and classes from one project to another

FIGURE 9-2: The Discover page on GitHub with repository recommendations.

GitHub employs machine learning techniques to generate a list of recommendations based on repositories you've starred, contributed to, and viewed. The recommendations also factor in the people you follow on GitHub. Chapter 17 goes into more detail on these actions and why they're important.

TIP To improve the recommendations, be mindful of the repositories you star and the people you follow.

Trending repositories

The next section on the Explore page includes a list of the top 25 trending repositories. Click the heading to visit `https://github.com/trending`, a page where you can discover repositories and people that are trending across GitHub.

TIP Why only 25 trending repositories? To have more than 25 trending repositories would dilute what it means to be trending. Also, it takes a lot to compute trending repositories, so limiting it keeps the cost low.

This page lets you filter trending repositories based on the primary language of the repository. If you're interested in the trending JavaScript libraries, click the Other Languages button and select JavaScript.

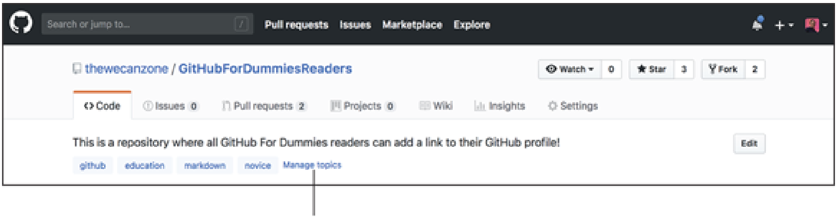
The Trending button at the top of the page defaults to today, but you can click it to see what's been trending for the week and month.

To determine what's trending, GitHub looks at a variety of data points, such as stars, forks, commits, follows, and pageviews. GitHub weighs these data points appropriately and factors in how recent the events were, not just total numbers.

Exploring topics

The Popular topics section lists the most popular topics on GitHub. A *topic* is a user-applied category for a repository. A topic gives people more information about what a repository is about. A repository owner can specify multiple topics for a repository. The concept is very similar to tags or issue labels.

Figure 9-3 shows the list of topics for the `thewecanzone/GitHubForDummiesReaders` repository. Admins for the repository see a Manage topics link that lets them add or remove topics for the repository.

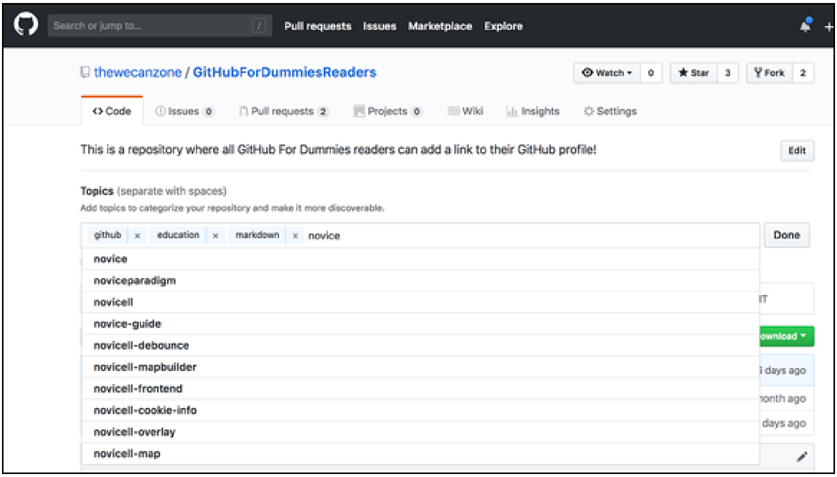

Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 3
Fork 2
Code
Issues 0
Pull requests 2
Projects 0
Wiki
Insights
Settings
Edit
github
education
markdown
novice
Manage topics

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

Manage topics link

FIGURE 9-3: List of topics for a repository.

When you add topics, you type a portion of a topic name, and GitHub offers suggestions based on the topics that others use within GitHub. Figure 9-4 shows an example where we type novice, and GitHub suggests other topics with the word novice in them.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
thewecanzone / GitHubForDummiesReaders
Watch 0
Star 3
Fork 2
Code
Issues 0
Pull requests 2
Projects 0
Wiki
Insights
Settings
Edit
Topics (separate with spaces)
github ×
education ×
markdown ×
novice
Done
novice
noviceparadigm
novicell
novice-guide
novicell-debounce
novicell-mapbuilder
novicell-frontend
novicell-cookie-info
novicell-overlay
novicell-map

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

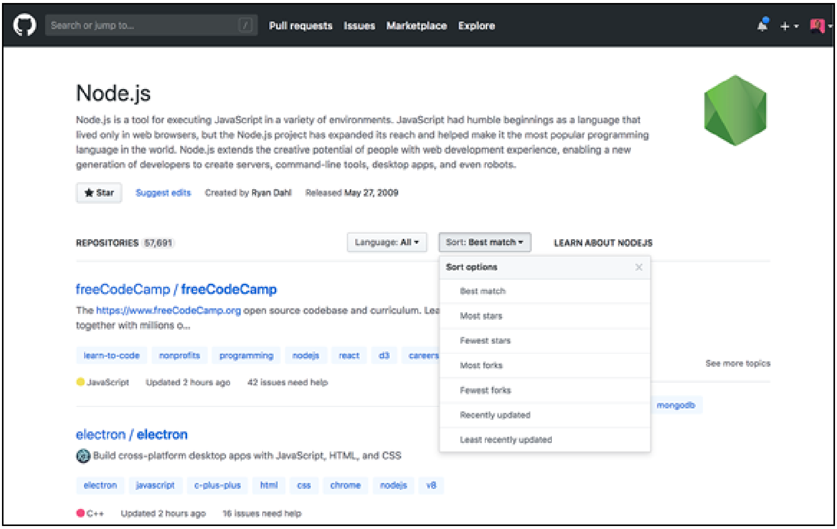
Add topics to categorize your repository and make it more discoverable.

FIGURE 9-4: A list of suggested topics.

Visit the Topics page at `https://github.com/topics` to see the most used topics on GitHub.

Clicking a specific topic is a great way to explore repositories related to a subject that you're interested in. For example, if you're interested in exploring `Node.js` repositories, you can navigate to `https://github.com/topics/nodejs`.

Topics are open-ended in that people can apply any topic they want to a repository. However, popular topics are often curated. For example, the `Node.js` topic page has a description and logo (see Figure 9-5). Figure 9-5 also shows the many options for sorting the topic's repositories via the expanded Sort button.


Node.js

Node.js is a tool for executing JavaScript in a variety of environments. JavaScript had humble beginnings as a language that lived only in web browsers, but the Node.js project has expanded its reach and helped make it the most popular programming language in the world. Node.js extends the creative potential of people with web development experience, enabling a new generation of developers to create servers, command-line tools, desktop apps, and even robots.

Created by Ryan Dahl Released May 27, 2009

freeCodeCamp / freeCodeCamp

The https://www.freeCodeCamp.org open source codebase and curriculum. Lea together with millions o...

JavaScript Updated 2 hours ago 42 issues need help

electron / electron

Build cross-platform desktop apps with JavaScript, HTML, and CSS

C++ Updated 2 hours ago 16 issues need help

Search or jump to...
Pull requests
Issues
Marketplace
Explore
★ Star
Suggest edits
REPOSITORIES 57,691
Language: All ▾
Sort: Best match ▾
LEARN ABOUT NODEJS
Sort options
Best match
Most stars
Fewest stars
Most forks
Fewest forks
Recently updated
Least recently updated
learn-to-code
nonprofits
programming
nodejs
react
d3
careers
See more topics
mongodb
electron
javascript
c-plus-plus
html
css
chrome
nodejs
v8

FIGURE 9-5: Topic page for NodeJS.

The topic descriptions and logo are themselves specified in an open source repository. Anyone can suggest edits for an existing topic description. Anyone can propose a topic description and logo for a new topic.

The descriptions and logos are located in the `https://github.com/github/explore` repository. For example, the topic logo and description for the Node.js repository is located in this directory `https://github.com/github/explore/tree/master/topics/nodejs`.

Exploring Marketplace apps

The next section of the Explore page lists popular Marketplace apps. You can add these apps to a repository to enable more capabilities and improve the workflow when working with the repository.

Chapter 15 covers the GitHub Marketplace in more detail.

Exploring Events

The Events section lists a selection of upcoming GitHub affiliated events, such as GitHub Universe (GitHub's yearly flagship conference), GitHub Satellite (smaller GitHub community events hosted around the world), and others.

Check this page often to find out about future events, which can be a great way to connect with the larger GitHub community.

Exploring collections

The next three sections are curated collections. At the time of writing, they are currently Front-end JavaScript frameworks, Learn to Code, and Probot Apps.

Each collection is fully curated by a human. A collection may contain a list of web pages and repositories. The goal is to be a great starting point for learning a particular subject in depth by listing websites for further reading and repositories with related code.

For example, to edit The Learn to Code collection, suggest an edit to this file in the `github/explore` repository:

```
https://github.com/github/explore/blob/master/collections/
   learn-to-code/index.md.
```

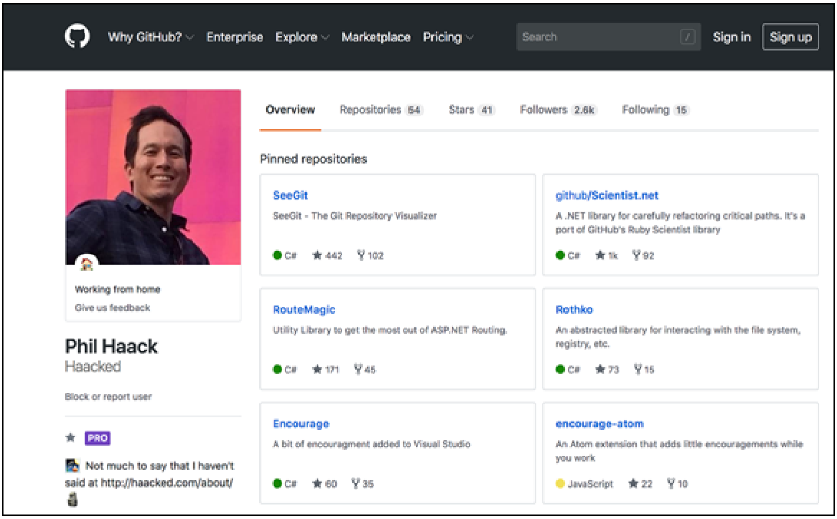
Getting by with help from your friends

Exploring what others on GitHub are up to is a great way to discover interesting new repositories. Whether it's your friends or other people you admire, be sure to pay attention to what they're up to on GitHub. One way to do so is through stars.

TIP As you explore repositories on GitHub, be sure to star the ones that pique your interest. To see your starred repositories, go to `https://github.com/stars`. Starring a repository not only is a good way to bookmark a repository for later exploration, but it also is a nice way to show a repository some recognition.

On the bottom left of the stars page is a grid of avatars for your friends on GitHub. Click a friend to see the repositories she's starred. Starred repositories often surfaces interesting repositories you may not have otherwise noticed.

You can also visit the profile page for your friends or other developers that you admire and look at their pinned repositories. *Pinned repositories* are repositories that a person explicitly chooses to feature. Figure 9-6 shows the pinned repositories for Phil, one of the authors of this book.



Working from home
Give us feedback

Phil Haack
Haacked

Block or report user

Not much to say that I haven't said at http://haacked.com/about/

SeeGit
SeeGit - The Git Repository Visualizer
C# ★ 442 102

github/Scientist.net
A .NET library for carefully refactoring critical paths. It's a port of GitHub's Ruby Scientist library
C# ★ 1k 92

RouteMagic
Utility Library to get the most out of ASP.NET Routing.
C# ★ 171 45

Rothko
An abstracted library for interacting with the file system, registry, etc.
C# ★ 73 15

Encourage
A bit of encouragment added to Visual Studio
C# ★ 60 35

encourage-atom
An Atom extension that adds little encouragements while you work
JavaScript ★ 22 10

FIGURE 9-6: Profile page for Phil, one of the authors.

Finding Places to Contribute

Some of you may be ready to move beyond just using open source code and are ready to hone your skills by contributing back to open source. How do you find a project to contribute to?

Exploring interesting repositories on GitHub comes into play when looking for a place to contribute. The only difference is that you'll have to narrow your exploration down even more when looking for a place to contribute.

The first thing to ask yourself is what type of repository do you want to contribute to. Answering this question can help narrow and guide your search.

There are many valid reasons for contributing to open source. One or more of the following may apply:

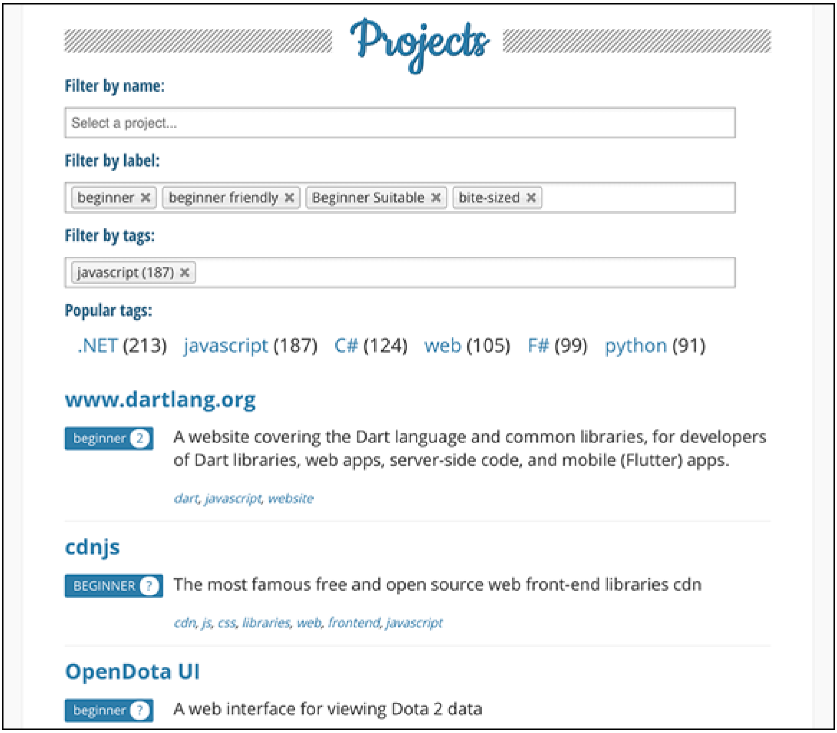
- I want to get involved in OSS in general and learn how to contribute.

- I want to help improve a project that I use in my day-to-day work.
- I want to support a project that does good in the world.
- I want to expand my skills by working in a technology I don't normally get to.
- I'm just bored and want to contribute to something cool.

No matter your reason for contributing to open source, a good place to start when approaching a new repository is to look for low-hanging fruit. Many repositories have some manner to indicate issues that would be great for first-time contributors. Often, they apply a label, such as `help-wanted` or `up-for-grabs`.

In fact, some websites scour issues labeled as such and make them searchable to those looking to get started with open source. The Up For Grabs website at `https://up-for-grabs.net` is one example.

The site has a search tool for filtering issues by project, label, and tag. Figure 9-7 shows an example where we're looking at all projects that have a `beginner` label and are tagged with `javascript`.



Projects

Filter by name:

Select a project...

Filter by label:

beginner × | beginner friendly × | Beginner Suitable × | bite-sized ×

Filter by tags:

javascript (187) ×

Popular tags:

.NET (213) javascript (187) C# (124) web (105) F# (99) python (91)

www.dartlang.org

beginner 2 A website covering the Dart language and common libraries, for developers of Dart libraries, web apps, server-side code, and mobile (Flutter) apps.

dart, javascript, website

cdnjs

BEGINNER ? The most famous free and open source web front-end libraries cdn

cdn, js, css, libraries, web, frontend, javascript

OpenDota UI

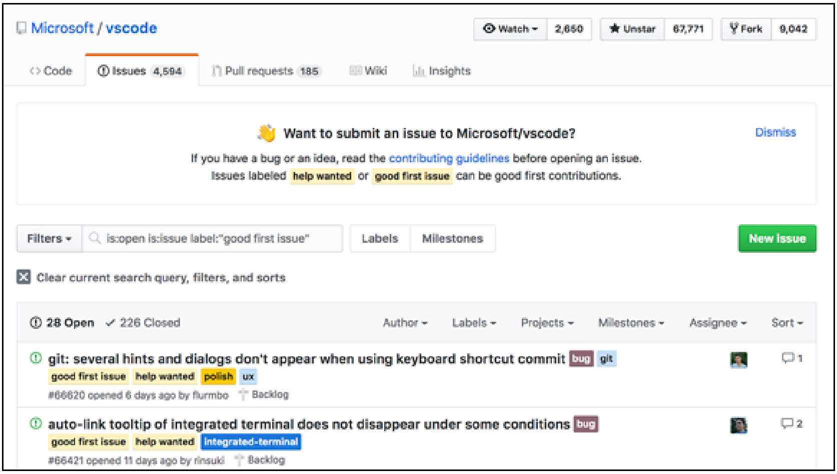
beginner ? A web interface for viewing Dota 2 data

FIGURE 9-7: Up for grabs filter.

Several repository results are displayed along with the label that matched the filter. Click the label for the repository result to see the specific issues for that repository with that label. Filtering issues is a good way to find a concrete issue to work on as a beginner.

A GitHub topic named `help-wanted`, which is at page, `https://github.com/topics/help-wanted`, lists repositories that are looking for help. When you visit a project that needs help, visit the issues page to look for potential issues you can help with.

For example, Figure 9-8 shows the issue tracker for the Visual Studio Code open source project. At the top is a message that mentions the project's contribution guidelines and a helpful tip to look for issues labeled help wanted or good first issue. Figure 9-8 shows the issues filtered by the `good first issue` label.


Microsoft / vscode
Watch 2,650
Unstar 67,771
Fork 9,042
Code
Issues 4,594
Pull requests 185
Wiki
Insights
Dismiss
Filters
Labels
Milestones
New issue
28 Open
✓ 226 Closed
Author
Labels
Projects
Milestones
Assignee
Sort
1
2

Want to submit an issue to Microsoft/vscode?

If you have a bug or an idea, read the contributing guidelines before opening an issue.
Issues labeled help wanted or good first issue can be good first contributions.

is:open is:issue label:"good first issue"

☒ Clear current search query, filters, and sorts

git: several hints and dialogs don't appear when using keyboard shortcut commit bug git
good first issue help wanted polish ux
#66620 opened 6 days ago by flurmbo Backlog

auto-link tooltip of integrated terminal does not disappear under some conditions bug
good first issue help wanted integrated-terminal
#66421 opened 11 days ago by rinsuki Backlog

FIGURE 9-8: Visual Studio Code Issues filtered by the `good first issue` label.

Surveying a Project for Contribution

Suppose that you decided that you want to contribute to a particular project. What are the next steps? For this example, we look at the Visual Studio Code repository at `https://github.com/`

`microsoft/vscode`.

Reading the CONTRIBUTING guide

As the suggestion on the Issues page notes, read through the contributing guidelines first.

TIP The contributing guidelines message displayed on the Issues page is controlled by a convention. Add a file named `CONTRIBUTING.md` to the root of the repository, a folder named `docs`, or a folder named `.github` in order to specify contributing guidelines for a repository.

Visual Studio Code's contributing guidelines are located at `https://github.com/Microsoft/vscode/blob/master/CONTRIBUTING.md`.

Like many contributing guides, the guide provides a high-level overview of how to make contributions to the project. It notes that contribution is more than just writing code. To emphasize that philosophy, the guide starts by letting you know where to ask questions about the project. The guide address the key topics:

- Where to ask questions
- Where to provide feedback
- Where and how to report issues
- Details about their automated issue management
- A link to a guide on how to contribute code

This format is pretty common for contributing guides. Sometimes, a smaller project will also include how to contribute code in the main document, but for larger or more complex projects, contributing code is a big topic.

Reading the contributing code guide

The VS Code project keeps its code contribution guide in a wiki located at `https://github.com/Microsoft/vscode/wiki/How-to-Contribute`.

REMEMBER If you plan to contribute code, read through this guide carefully.

The guide has the key information you'll want to code on the project and is structured in a fashion pretty common among open source projects:

- Building the code
- Running the code
- Debugging the code
- Running the automated tests
- Running automated code analysis, such as linting

The next section covers the code contribution workflow:

- Following the branching strategy
- Creating pull requests
- Packaging code for distribution

Reading the code of conduct

The contributing guide focuses on the mechanics of making a contribution. Many projects now also include a `CODE_OF_CONDUCT.md` file. This file lays out expectations for those who participate in the project. It sets the behavioral norms for the project and a resolution process in cases where violations of the norm occur.

VS Code uses the Microsoft Open Source Code of Conduct located at `https://opensource.microsoft.com/codeofconduct`.

TIP Adding a code of conduct to your own repository is easy. If you add a new file on GitHub using the browser and name the file `CODE_OF_CONDUCT.md`, GitHub displays a code of conduct selection drop-down list with two choices: the

Contributor Covenant and the Citizen Code of Conduct.

You can also visit the community page for any public repository to add a code of conduct. Go to the community page by appending `/community` to the end of the repository URL. For example, type `https://github.com/microsoft/vscode/community` to see w the project has a code of conduct. If it doesn't, the page gives you the opportunity to propose one.

Setting Contributor Expectations

After you’ve read the code of conduct and contribution guides, you’re ready to dive in and make some contributions. Now’s a good time to make sure that you have clear expectations for how the process will work.

REMEMBER These general guidelines for contributors may not be spelled out in a contributing guide, but are more the result of collective wisdom gathered from working in open source for a long time.

They won’t fix every issue

Many, if not most, projects have limited resources, and their priorities may not align with your priorities. It’s important to keep all that in mind when you file an issue. On the one hand, opening a good issue takes time and energy. A well-written issue that describes a problem clearly with clear steps to reproduce the issue is very valuable to a project, so it’s understandable that people who write issues feel invested in them.

REMEMBER On the other hand, the fact that an issue may not be fixed is why we note that writing an issue is a contribution. It’s not an exchange of one thing for another, but it’s a gift. And as such, you can’t expect much in return. Good project maintainers will thank you for filing an issue and note whether they know of any workarounds at minimum, but they do not owe you a fix. Do not take it personally if an issue you file is labeled as `won't fix`.

They won’t merge every pull request

Submitting a pull request is the culmination of a lot of work. In order to submit a proper pull request, a developer had to spend the time to get the code working on his own machine, understand the code well enough to make a change, write the change, and then submit it.

Being disappointed when the repository maintainers then reject the pull request is understandable. Remember, though, they’re under no obligation to accept your pull request. Yes, you put a lot of time into the pull request, but they’ll have to own the code change for the lifetime of the project.

REMEMBER Having a pull request rejected should be a very rare occurrence. You and the project maintainers can do a lot beforehand to avoid a rejected pull request. The first step starts with communication. Before you start work to resolve an issue, comment on the issue with your intentions. Indicate your general plan of attack. Make sure that someone related to the project responds and agrees with your general approach. This communication reduces the likelihood that you go off and work on something in a matter completely contrary to the project.

Don’t stop your communication there, though. Keep it going. As soon as you have a commit or two, push it to a pull request and make sure the pull request is clearly marked as a work in progress. Marking the pull request as a work in progress lets you continue to get feedback as you go and ensure that you’re on the right path. It minimizes the time spent going down the wrong path.

Finally, make sure that you’re following all their contribution guidelines, style guidelines, and so on. This is their project, and they will own the code if they choose to merge your PR in. This is not the time and place to try to advocate for your way of working or your personal code style.

They don’t owe you anything

One of the biggest challenges of being a maintainer of a popular open source project is the sense of entitlement expressed by many people who use the software.

REMEMBER In most cases, the maintainers are all volunteers working on the project on the side. This isn't always the case. Sometimes, they're paid employees working on an open source project. But chances are, you are not the one paying them. They don't owe you a feature. They don't owe you a bug fix. They don't owe you anything. And you should treat them accordingly.

Of course, the maintainers of a well-run project will try to go out of their way to accommodate people who participate in a project. It's only natural that they want most people involved to be happy. Open source projects benefit from mind share and more contributors, so it's often in their best interests to not take a hard line and try to get you that feature if it fits their roadmap and priorities and isn't too much trouble. So it's not wrong to ask for things. It's not wrong to make a strong case for things you want. But at the end of the day, you don't pay them, and they don't owe you a thing.

Keeping Tabs on a Project

When you're a contributor to a project, it's good to keep tabs on how the project is doing. One way is through GitHub notifications. Chapter 1 covers how to manage notification settings. GitHub can send notifications for new issues, new comments on pull requests, and so on. It's important to manage your notification settings based on your interest level in the project.

While GitHub may be the hub of open source, a lot goes on outside of GitHub related to an open source project. You may want to look at the places where a project lives outside of GitHub. For example, many projects have a Twitter account you can follow. Some projects have a public Slack instance where you can chat with the maintainers and others who use the project. You also may want to read the blogs of the maintainers of those projects to keep on top of the latest developments. Many projects and project maintainers still publish RSS feeds, an ancient technology that makes it possible to keep up with a site.

Chapter 10

Starting Your Own OSS

IN THIS CHAPTER

- **Setting up a repository for OSS**
- **Managing OSS workflows**
- **Ending an OSS project**

There comes a point in many developers lives when they have their own code to share with the world. Many different approaches and motivations may drive a person to share code. In some cases, a person may write some code they think others will find interesting and just “toss it over the wall” without any intent to take contributions back. On the other end is the case where someone wants to start a real movement with large numbers of contributors.

Whatever your motivation, the only requirement for open source software is to choose a license that complies with the Open Source Definition (OSD). You can find the definition at `https://opensource.org/osd`. The Open Source Initiative (OSI) has a process for approving licenses, and its site lists a plethora of licenses.

But for the life of most open source software, choosing a license is just a starting point. A lot more goes with managing an open source project. In this chapter, we cover what it means to start an open source project on GitHub, maintain it, and if it comes to it, sunset it.

Creating an Open Source Repository

When you create a repository with the intent to make it open source, it’s common to make the repository public and select a license during the repository creation process.

Sometimes it is preferable to defer those choices to later. Perhaps you want to set up your repository right before you make it public to the world. Or you may want to spend more time thinking about the license. The following sections focus on the scenario of turning an existing private repository into an open source one. We assume that you've already created a repository with a `README.md` file, but didn't make the repository public nor did you choose a license. (If you haven't, see Chapter 3.)

Adding a license

The only requirement that software has to be considered open source is to have an open source license. GitHub makes adding a license to an existing project easy.

Figure 10-1 shows the home page for a private repository without a license. To add a license:

1. **Click the Create new file button.**

   After you click the button, GitHub prompts you to name the file.

2. **Type your filename.**

   For this example, we name ours `LICENSE`. GitHub notices you are attempting to create a license file and helpfully adds a button to the right labeled Choose a license template, as shown in Figure 10-2.

3. **Click the Choose a license template button to see a list of license templates.**

   The three most common licenses used on GitHub are listed first in bold, as shown in Figure 10-3. Chapter 3 provides a brief guide on choosing a license, but this page has a link to a more detailed guide to help you pick which license best fits your project.

4. **After you know which license you want, click the license to view information about the license.**

   This page includes a high-level summary with bullet points about the key traits of the license. This overview helps you gain an understanding of the license without having to wade through all the legalese.

Of course, the page also presents the full text of the license for those who relish legalese. On the right are fields required by the license.

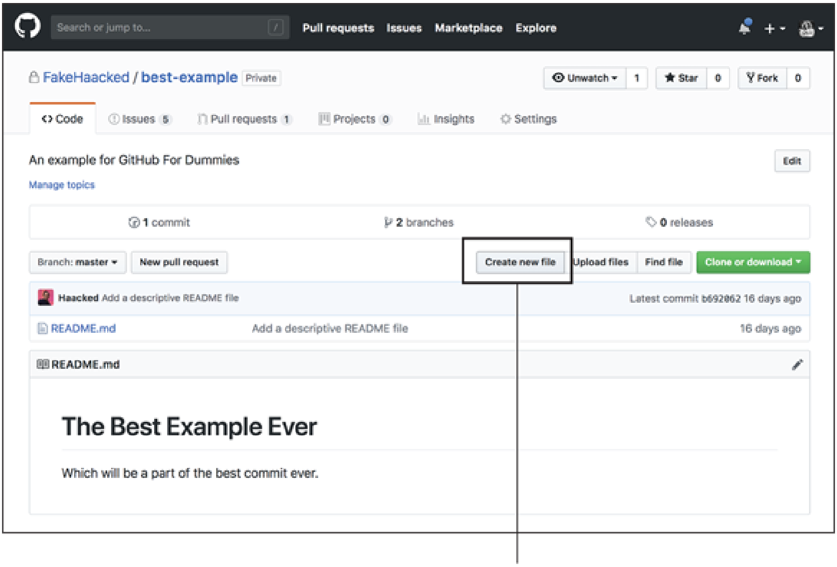
5. **Enter your information in the fields to customize the license to your situation.**

   You can see an example of this page in Figure 10-4.

6. **After you're satisfied with everything, click the Review and submit button.**

   You return to the file creation page with the text of the license filled in.

7. **Scroll down to enter your commit message and click Commit new file to create the license file.**


Search or jump to…
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Private
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Insights
Settings
Edit
Manage topics
1 commit
2 branches
0 releases
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
Haacked Add a descriptive README file
Latest commit b692062 16 days ago
README.md
Add a descriptive README file
16 days ago
README.md
Create new file button

An example for GitHub For Dummies

The Best Example Ever

Which will be a part of the best commit ever.

FIGURE 10-1: Home page for our soon-to-be open source repository.

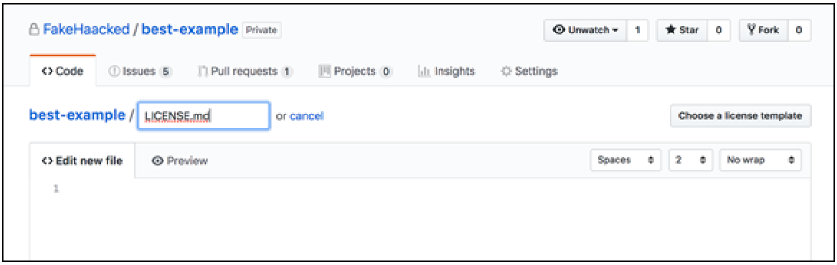
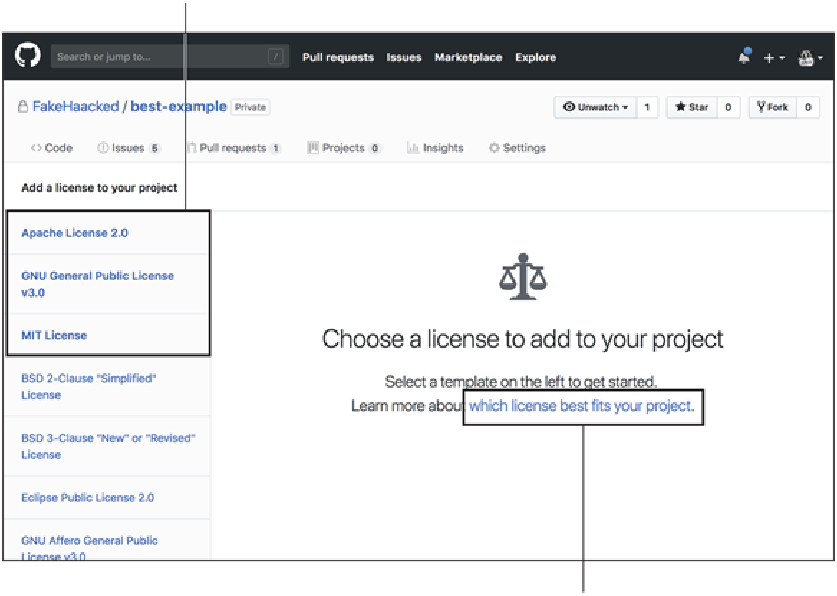

FakeHaacked / best-example Private
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Insights
Settings
best-example / LICENSE.md or cancel
Choose a license template
Edit new file
Preview
Spaces
2
No wrap
1

FIGURE 10-2: Choose a license template button magically appears.

The three most popular licenses on GitHub

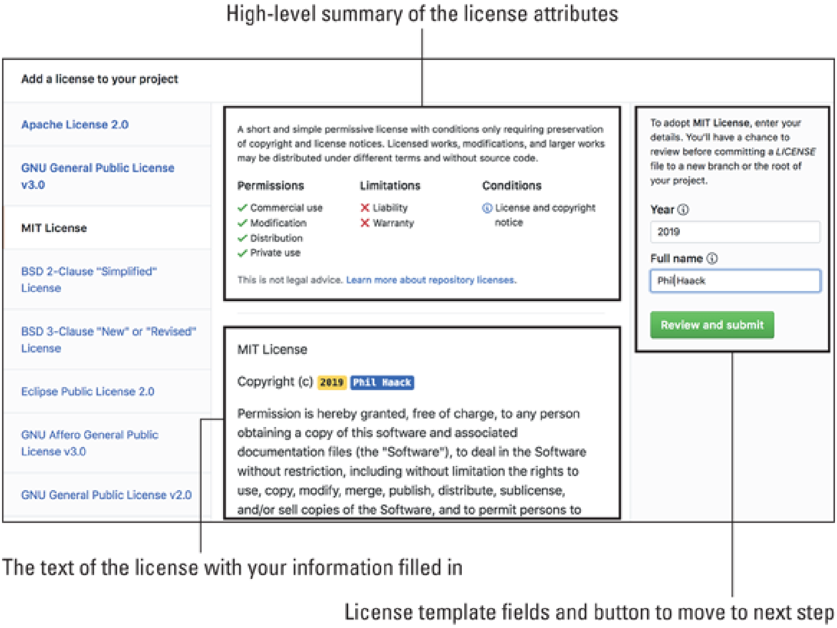

Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example Private
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Insights
Settings
Add a license to your project
Apache License 2.0
GNU General Public License v3.0
MIT License
BSD 2-Clause "Simplified" License
BSD 3-Clause "New" or "Revised" License
Eclipse Public License 2.0
GNU Affero General Public License v3.0

Choose a license to add to your project

Select a template on the left to get started.
Learn more about which license best fits your project.

Link to more information on choosing a license

FIGURE 10-3: The license chooser page.


High-level summary of the license attributes

Add a license to your project

Apache License 2.0

GNU General Public License v3.0

MIT License

BSD 2-Clause "Simplified" License

BSD 3-Clause "New" or "Revised" License

Eclipse Public License 2.0

GNU Affero General Public License v3.0

GNU General Public License v2.0

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	ⓘ License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. Learn more about repository licenses.

MIT License

Copyright (c) 2019 Phil Haack

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to

To adopt **MIT License**, enter your details. You'll have a chance to review before committing a *LICENSE* file to a new branch or the root of your project.

Year ⓘ: 2019

Full name ⓘ: Phil Haack

The text of the license with your information filled in

License template fields and button to move to next step

FIGURE 10-4: The license information page.

Adding contributor guidelines

Another useful document to include in your repository is a file named `CONTRIBUTING.md`. This file provides information on how to contribute to your project. It should include information such as where people should ask questions and where to provide feedback.

Chapter 9 provides more details on what a contributor can expect to find in a `CONTRIBUTING.md` document. That chapter is a good place to start for your own project's contribution guidelines.

REMEMBER A robust `CONTRIBUTING.md` document will save you and your future contributors a lot of headache and time. Be sure to revisit it from time to time to make sure it's up to date.

Adding a code of conduct

A code of conduct makes behavioral expectations clear for everyone who participates with your project, whether they contribute code, comment on an issue, or otherwise interact with others on your repository. A code of conduct sets the tone for the type of open and welcoming environment you want to foster for your project.

To add a code of conduct, follow the same process as you do when adding a license, but name the file `CODE_OF_CONDUCT.md`. (See the "Adding a license" section, earlier in this chapter.) This process is covered in Chapter 9 as well.

Making a Repository Public

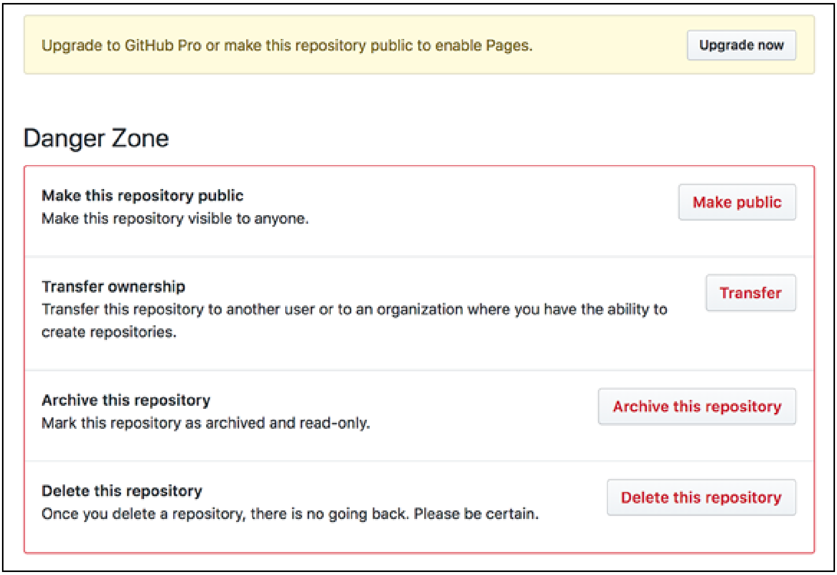
At this point, your repository has the basic items it needs to go public. To change a repository from private to public:

1. **Visit the Settings page for your repository.**
2. **Scroll to the bottom of the Settings page to reach the Danger Zone, as shown in Figure 10-5.**
3. **Click the Make public button to initiate making a repository public.**

   WARNING This step is a potentially dangerous operation because you may inadvertently expose information to the world you'd rather keep private. So make sure you really are ready to make this repository public. If you created it with the intention of making it public from the beginning, the repository shouldn't contain any secrets.

   GitHub displays a confirmation dialog after you click the Make public button asking you to type the name of the repository to confirm this change, as seen in Figure 10-6.
4. **Type the repository name and then click the I understand, make this repository public button to make the repository public.**

   Thee setting page now has a section that will come in handy as you manage your new open source project.



Upgrade to GitHub Pro or make this repository public to enable Pages.

Upgrade now

Danger Zone

Make this repository public
Make this repository visible to anyone.

Make public

Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories.

Transfer

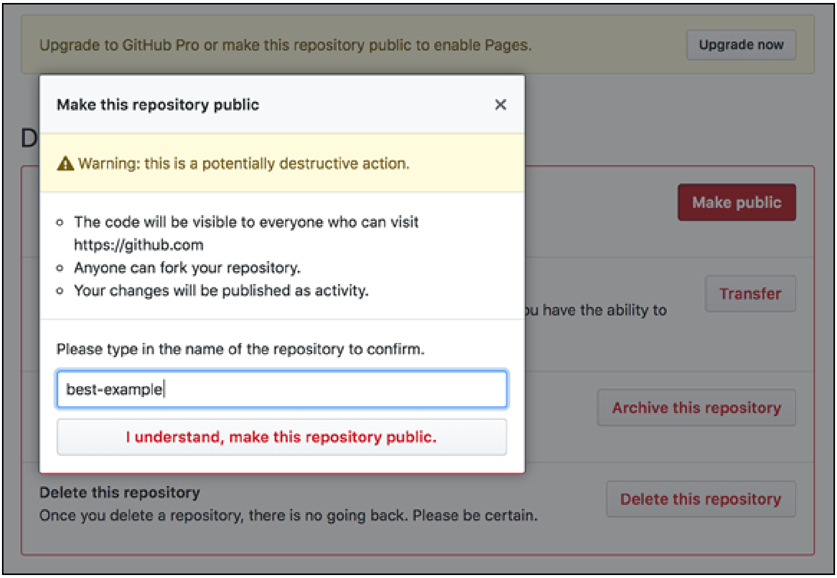
Archive this repository
Mark this repository as archived and read-only.

Archive this repository

Delete this repository
Once you delete a repository, there is no going back. Please be certain.

Delete this repository

FIGURE 10-5: Danger zone: where you can make a private repo public.



Make this repository public

⚠ Warning: this is a potentially destructive action.

- The code will be visible to everyone who can visit https://github.com
- Anyone can fork your repository.
- Your changes will be published as activity.

Please type in the name of the repository to confirm.

best-example

I understand, make this repository public.
Upgrade now
Make public
Transfer
Archive this repository
Delete this repository

FIGURE 10-6: Confirmation to convert a private repo public.

Enforcing a Code of Conduct

A *code of conduct* is a great way to communicate behavioral expectations for a community. But a code of conduct isn't enough to create a healthy community on its own; you have to enforce it.

WARNING It would be wonderful if you never had to enforce your code of conduct. Most people who participate in your project will be helpful and kind. However, as your project grows and more people get involved, it's almost inevitable that you'll have to step in.

Responding with kindness

Keep in mind that even when someone acts in a matter that doesn't fit in the spirit of your code of conduct, they may not necessarily be malicious or a bad actor. Maybe they're having a bad day, and they took it out on your repository. That doesn't excuse their behavior, but it is relatable. We've all done it.

TIP If someone makes an off color or angry comment in an issue in your repository that goes against the type of community you want to foster, it's often disarming to reply back with a kind response that expresses empathy for their situation, but is clear that the comment doesn't meet community guidelines. Be specific in how it does not meet the guidelines and if you can, suggest an alternative approach they could have taken.

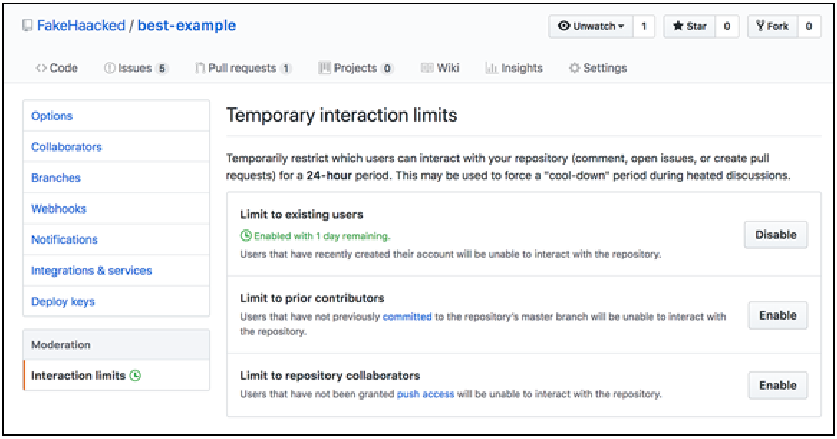
For example, if someone comments on the repository that the maintainer must hate other people because they haven't fixed a particular bug, you can respond with something like "Personal attacks are not welcome in this repository. We are all volunteers trying to work on this in our spare time. I understand that you are frustrated. Instead of a personal attack, perhaps write about how the bug affects you and why that is frustrating to you."

REMEMBER But always remember, you don’t owe anybody anything. You’re not there to be a punching bag for people. If they’re abusive or if you’re simply tired of a thousand paper cuts of negative comments, rather than respond while angry, take a break. Walk away. Ask for help. Your own mental well-being comes first.

Leveraging the ban hammer

It would be nice if a kind admonishment worked in every situation (see preceding section), but sometimes tempers flare. We’re all human, and sometimes we lose our temper. So will people who participate in your repository. This situation is where temporary interaction limits come in handy.

At the very bottom of the Settings page for your public repository (this setting doesn’t exist for a private repository) is a moderation section with a link to Interaction limits. Click this link to see the set of limits available. Figure 10-7 shows this page after enabling the Limit to existing users option.


FakeHaacked / best-example
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Wiki
Insights
Settings
Options
Collaborators
Branches
Webhooks
Notifications
Integrations & services
Deploy keys
Moderation
Interaction limits
Disable
Enable
Enable

Temporary interaction limits

Temporarily restrict which users can interact with your repository (comment, open issues, or create pull requests) for a 24-hour period. This may be used to force a "cool-down" period during heated discussions.

Limit to existing users
Enabled with 1 day remaining.
Users that have recently created their account will be unable to interact with the repository.

Limit to prior contributors
Users that have not previously committed to the repository's master branch will be unable to interact with the repository.

Limit to repository collaborators
Users that have not been granted push access will be unable to interact with the repository.

FIGURE 10-7: The set of interaction limits.

REMEMBER These limits are not permanent bans. They are cool-down periods. This tool is great to use when a discussion or an issue gets a bit too heated. It gives everyone time to cool down and prevents the discussion from running even further out of control.

Blocking users

Every once in a while, you may encounter a truly abusive person in your project. You've tried to kill them with kindness. You've tried the 24-hour cool-down periods, but they persist in being rude or antagonistic. It is time to block or report the user.

To block or report the user:

1. **Click the user's name next to their comment to visit their profile page.**

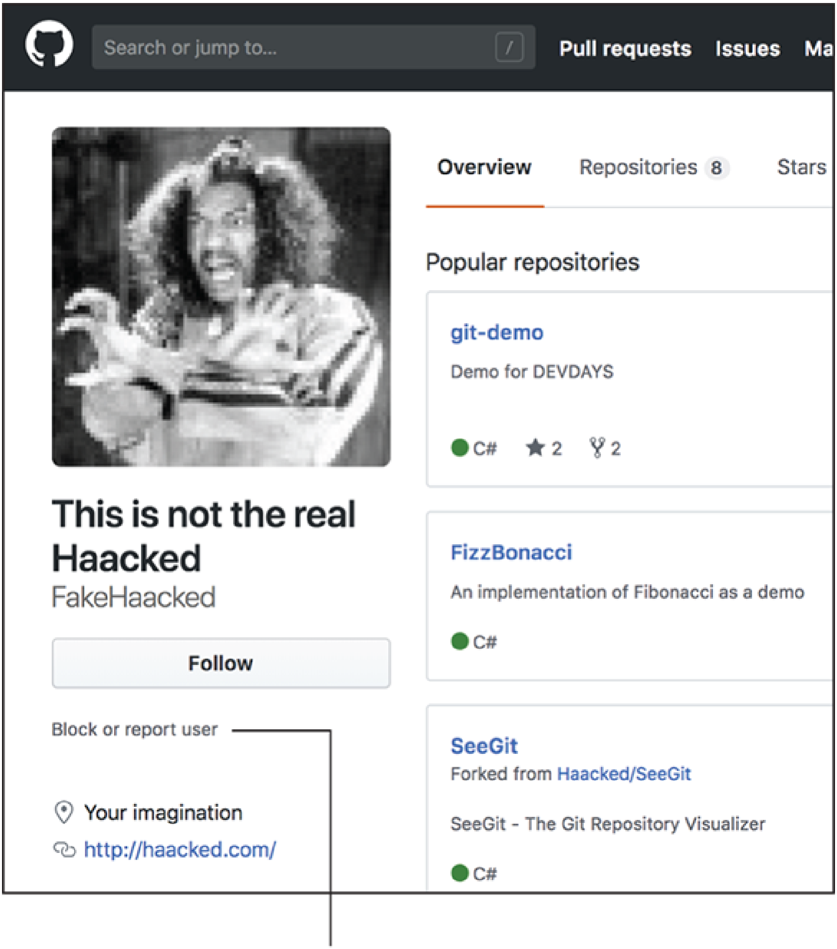
   Underneath their profile picture is a link to block or report the user, as shown in Figure 10-8.

2. **Click the link to block or report the user.**

   A dialog box with two options appears: Block user or Report abuse, as shown in Figure 10-9.

3. **Click Block user to immediately block the user.**

   Blocking a user denies a user from accessing your activity and repositories. It also prevents them from sending you notifications, such as when they @mention your username. They pretty much no longer exist as far as you are concerned, and vice versa. For more details on what happens when you block a user, visit `https://help.github.com/articles/blocking-a-user-from-your-personal-account/`.


Search or jump to...
/
Pull requests
Issues
Ma
Overview
Repositories 8
Stars
Follow
Block or report user

This is not the real Haacked
FakeHaacked

Your imagination
http://haacked.com/

Popular repositories

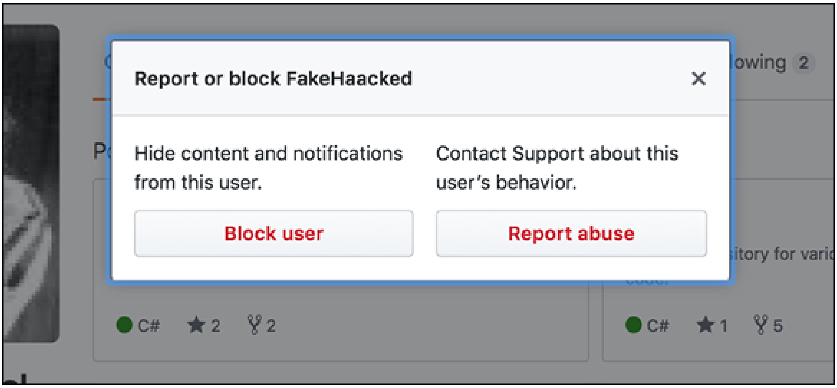
git-demo
Demo for DEVDAYS
C# ★ 2 2

FizzBonacci
An implementation of Fibonacci as a demo
C#

SeeGit
Forked from Haacked/SeeGit
SeeGit - The Git Repository Visualizer
C#

Link to block or report user

FIGURE 10-8: Profile page with link to block or report user.



Report or block FakeHaacked

Hide content and notifications from this user.

Contact Support about this user's behavior.

FIGURE 10-9: Dialog box with Block user and Report abuse buttons.

TIP

You can also block users, see a list of users you currently have blocked, and unblock users from your account Settings page by going to `https://github.com/settings/blocked_users`.

Writing a README.md File

The `README.md` file holds a special significance on GitHub. When you visit a repository's home page, GitHub displays the contents of the `README.md` rendered as Markdown on the home page. The `README.md` contents is what visitors to your repository will expect to see to learn more about your project.

Chapter 3 includes information about what should be included in a `README.md` file. It should also include links to your `CONTRIBUTING.md` file and your `CODE_OF_CONDUCT.md` file. After reading this file, a person should understand what your project does, how to get involved, and where to go to find out more information.

Writing Good Documentation

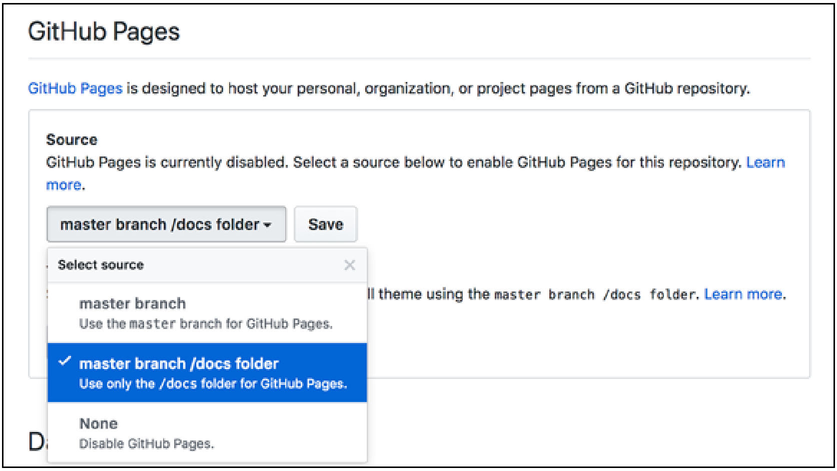
One of the most neglected aspects of open source is documentation. Writing good docs is a great way to distinguish your project from the rest. Also, documentation provides a nice way for others to dip their toes into open source. One of the authors got his start in open source by contributing docs to a project many, many years ago.

TIP GitHub supports a wiki feature, but we recommend using a relatively unknown feature, serving a documentation site from the `docs` folder in the master branch of your repository. This feature has many benefits, but one of the big ones is the ability to version your documentation along with the code that it documents.

Chapter 4 walks through building a GitHub Pages site. Serving your documentation from the `docs` folder is very similar to that process. The following steps assume that you have a repository that is not a GitHub Pages website. Chapter 3 walks through creating a regular repository if you need a refresher.

Make sure to commit a file named `index.md` to the `docs` folder in the master branch of your repository. This feature requires that there's already a `docs` folder before you can enable it. If you’re unclear about how to do that, Chapter 7 covers writing and committing code in detail.

Navigate to your repository settings and then scroll down to the GitHub Pages section. By default, the Source is set to None. Click the button and select the master branch /docs folder option as shown in Figure 10-10 and then click the Save button.



GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more.

ll theme using the `master branch /docs folder`. Learn more.

master branch /docs folder
Save
Select source
master branch
Use the master branch for GitHub Pages.
✓ master branch /docs folder
Use only the /docs folder for GitHub Pages.
None
Disable GitHub Pages.
D

FIGURE 10-10: Serving a GitHub Pages site from the docs folder.

You can now visit your documentation site at `https://{USERNAME}.github.io/{REPO-NAME}/`. To see an example of this in action, visit `https://fakehaacked.github.io/best-example/`.

As we cover in Chapter 4, you can select a theme and add a custom domain name if needed.

REMEMBER Don’t forget to add a link to the docs site to your README file so that people can find it.

Managing Issues

As the word gets out about your open source project, people will start to try it out. And at some point, people will open issues. Congratulations! When an issue is opened, it means someone out there cared enough to report a bug, ask a question, or make a feature request. This interest is a good thing for an open source project.

But if your project becomes very successful, the influx of new issues can start to get overwhelming. In the following sections, we cover some tips on managing incoming issues.

Labeling issues

GitHub provides a default set of issue labels when you create a website.

- bug
- duplicate
- enhancement
- good first issue
- help wanted
- invalid
- question
- ready for review
- won't fix

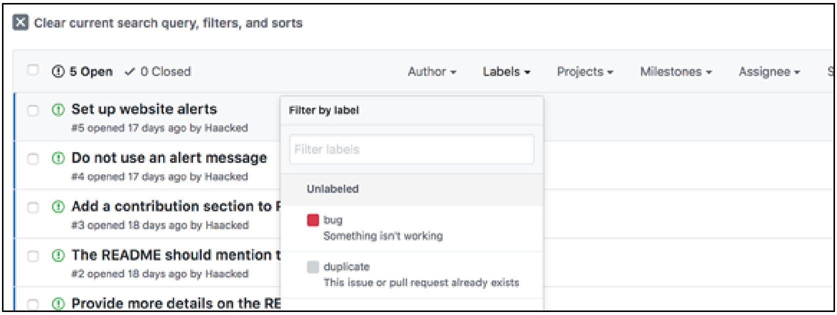
You can manage these labels by going to the Issues tab and clicking the Labels button. Make sure the set of labels makes sense for your project. Assigning labels helps you manage issues as they come in.

Triaging issues

As new issues are created, it's important to triage issues. The word *triage* comes from the medical field. Merriam-Webster defines triage as "the sorting of patients (as in an emergency room) according to the urgency of their need for care."

Triaging issues is similar to triaging in a hospital. *Triage,* when applied to issues, is the process of reviewing new issues and assigning labels to indicate what type of issue are they (bug report, feature request, and so on.), determining their priority, and assigning issues to people. In some cases, you may close issues you don't plan to address during triage.

TIP Every triaged issue should have some label applied. If necessary, you can even create a triaged label for this purpose. Assigning a triaged label to issues that you've triaged makes it possible to see all issues that haven't yet been triaged by using the unlabeled filter, as shown in Figure 10-11.


Clear current search query, filters, and sorts
5 Open
✓ 0 Closed
Author ▾
Labels ▾
Projects ▾
Milestones ▾
Assignee ▾
Filter by label
Filter labels
Unlabeled
bug
Something isn't working
duplicate
This issue or pull request already exists

Set up website alerts
#5 opened 17 days ago by Haacked

Do not use an alert message
#4 opened 17 days ago by Haacked

Add a contribution section to
#3 opened 18 days ago by Haacked

The README should mention
#2 opened 18 days ago by Haacked

Provide more details on the RE

FIGURE 10-11: Unlabeled issues filter.

To maintain a sense of sanity, it often makes sense to schedule triage on a regular cadence as opposed to doing it all the time as people create new issues.

Issue templates

It would be great if everyone who submits an issue understood how to write up a proper bug report or if they understood what makes for a good feature request. A bug report that just says "Your stuffs is the broken" is not too helpful to the repository maintainer.

To help foster good issues, you can set up issue templates that guide people filing issues on your repository to supply the information you need.

In the Settings page for your repository, in the Issues section in the main pane, is a big green button labeled Set up templates. Clicking that button takes you to a templates setup page where you can choose from a set of pre-existing issue templates or create your own, as shown in Figure 10-12.

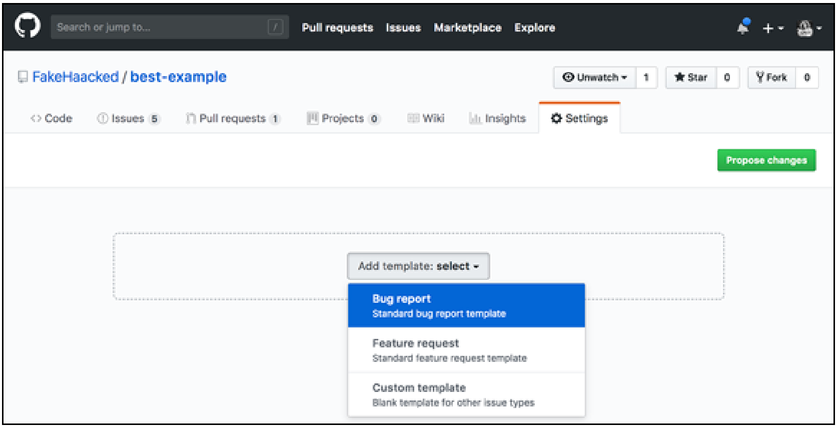

Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Wiki
Insights
Settings
Propose changes
Add template: select
Bug report
Standard bug report template
Feature request
Standard feature request template
Custom template
Blank template for other issue types

FIGURE 10-12: Choosing an issue template.

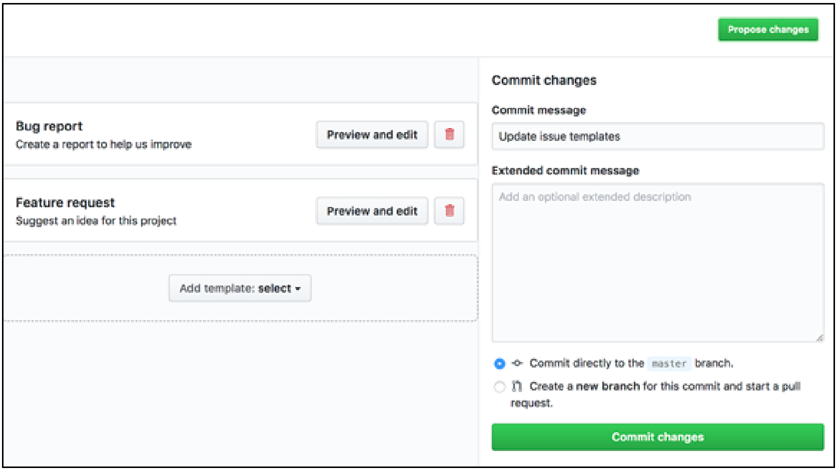
From the templates setup page, follow these steps:

1. **Select both the Bug report template and then select the Feature request template.**

   You should see them listed on the page. Note they're not yet active. They still need to be committed to the repository.

2. **Click the Propose changes button to enter a commit message for these templates, as shown in Figure 10-13.**
3. **Click the Commit changes button to commit these templates to the repository.**

   They are stored in a special `.github/ISSUE_TEMPLATE` folder. If you navigate to that folder, you should see two files, `bug_report.md` and `feature_request.md`. Take a look at the contents of those files to understand the structure of an issue template.


Propose changes
Bug report
Create a report to help us improve
Preview and edit
Feature request
Suggest an idea for this project
Preview and edit
Add template: select
Commit changes
Commit message
Extended commit message
Add an optional extended description
Commit changes

Commit message: Update issue templates

● Commit directly to the `master` branch.

○ Create a new branch for this commit and start a pull request.

FIGURE 10-13: Choosing an issue template.

You can create new templates by going through the previous steps or by manually adding a file to the `.github/ISSUE_TEMPLATE` directory with the same basic structure as the existing issue template files.

Now, when someone creates an issue on your repository, they first see a set of issue templates, as shown in Figure 10-14.

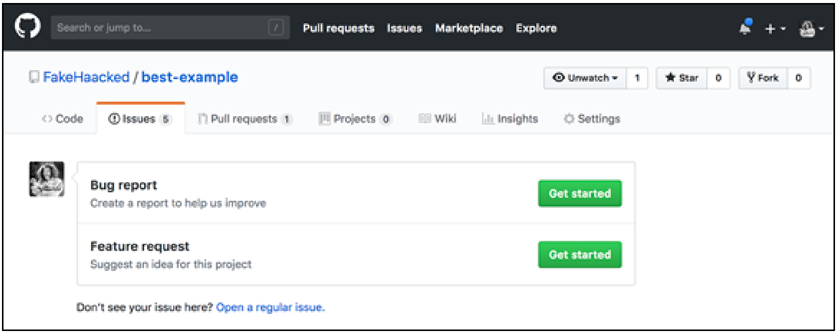
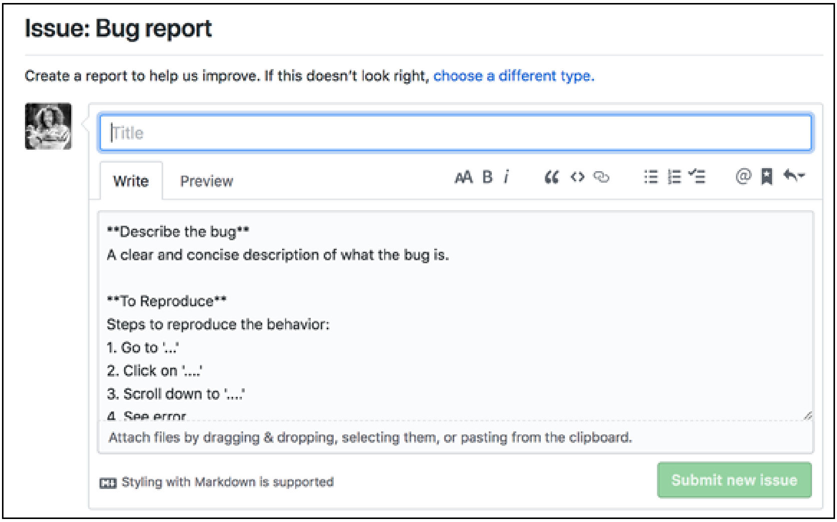

Search or jump to...
Pull requests
Issues
Marketplace
Explore
FakeHaacked / best-example
Unwatch 1
Star 0
Fork 0
Code
Issues 5
Pull requests 1
Projects 0
Wiki
Insights
Settings
Bug report
Create a report to help us improve
Get started
Feature request
Suggest an idea for this project
Get started
Don't see your issue here? Open a regular issue.

FIGURE 10-14: Choose an issue template to create an issue.

Clicking the Get started button on one of the templates then displays the issue creation form, but with the contents prepopulated with the template information. Figure 10-15 shows an issue prepopulated with the default Bug report template.



Issue: Bug report

Create a report to help us improve. If this doesn't look right, choose a different type.

```
**Describe the bug**
A clear and concise description of what the bug is.

**To Reproduce**
Steps to reproduce the behavior:
1. Go to '...'
2. Click on '....'
3. Scroll down to '....'
4. See error
```

Title
Write
Preview
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Submit new issue

FIGURE 10-15: A new issue with the Bug report template filled in.

The contents of the issue walks the issue creator through all the information expected of them. Of course, the person creating the issue can always choose to ignore the template and put anything they want in the issue.

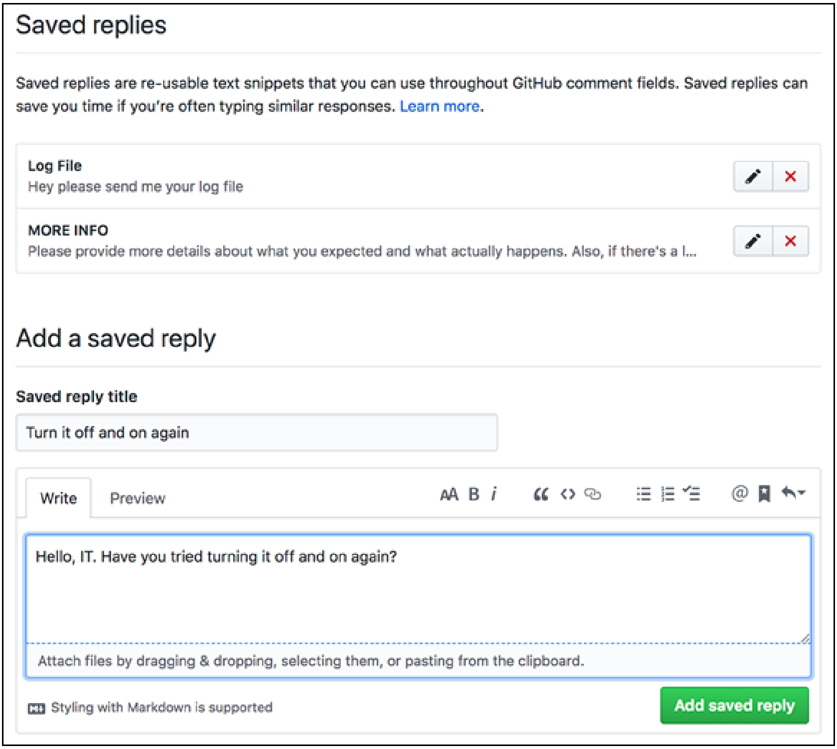
Saved replies

Often, as your repository grows in popularity and the issues start to flow in, you'll find yourself repeating responses over and over again. For example, if someone reports a bug, you may need them to supply a log file. If every bug report should include a log file, then you should mention that in an issue template.

But even if you do, many folks will forget to include it. This situation is where a saved reply can come in handy. A *saved reply* is a canned response you can use over and over again.

As of this writing, canned responses apply only to a user and are not specific to a repository. So you can create your own saved replies, but they will not immediately be available to other members of your repository.

Go to `https://github.com/settings/replies` to manage your list of saved replies. To create a new saved reply, fill in the form at the bottom, as shown in Figure 10-16.



Saved replies

Saved replies are re-usable text snippets that you can use throughout GitHub comment fields. Saved replies can save you time if you're often typing similar responses. Learn more.

Log File
Hey please send me your log file

MORE INFO
Please provide more details about what you expected and what actually happens. Also, if there's a l...

Add a saved reply

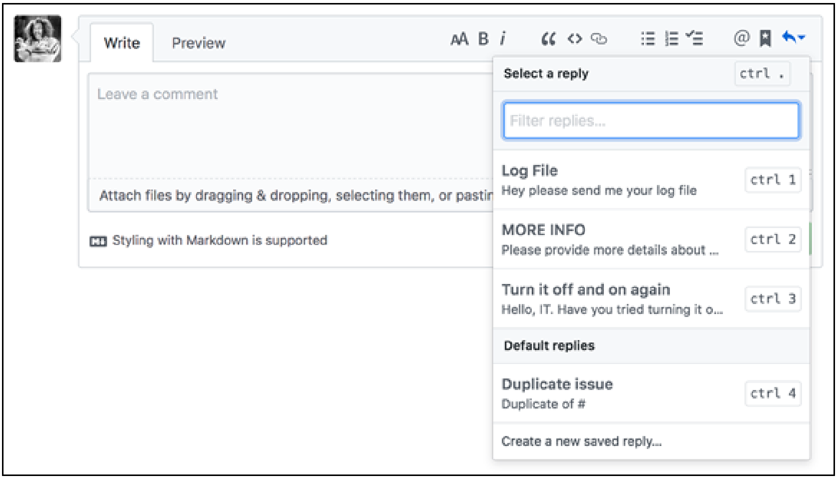
Saved reply title
Turn it off and on again

Hello, IT. Have you tried turning it off and on again?

Write
Preview
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Styling with Markdown is supported
Add saved reply

FIGURE 10-16: Creating a new saved reply.

To access the saved replies when responding to an issue or a pull request comment, click the arrow in the top right corner of the comment box, as shown in Figure 10-17. That brings up a list of your saved replies. Click the one you want to use to fill the comment box with that reply.


Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting them, or pastin
Styling with Markdown is supported
Select a reply
ctrl .
Filter replies...
ctrl 1
ctrl 2
ctrl 3
Default replies
ctrl 4
Create a new saved reply...

Log File
Hey please send me your log file

MORE INFO
Please provide more details about ...

Turn it off and on again
Hello, IT. Have you tried turning it o...

Duplicate issue
Duplicate of #

FIGURE 10-17: Selecting a saved reply in a comment box.

Ending Your Project

At some point in the life of a repository, you may want to step away from it. This break may happen for many reasons. Perhaps the project is no longer useful, having been supplanted by other better projects. Perhaps the project is a runaway success, but you don't have the time to give it the attention it deserves.

Whatever the reason, it's important to handle the end of your involvement with the repository with the same care you showed in starting it.

Archiving a project

Archiving a project is a good option when a project is no longer all that useful to others. Or even if it's still useful, but mostly complete, archiving could be a good option. *Archiving* a project indicates that the project is no longer actively maintained. The code is still available to the world, and people can fork and star your project, but nobody can create new issues, pull requests, or push code to an archived repository.

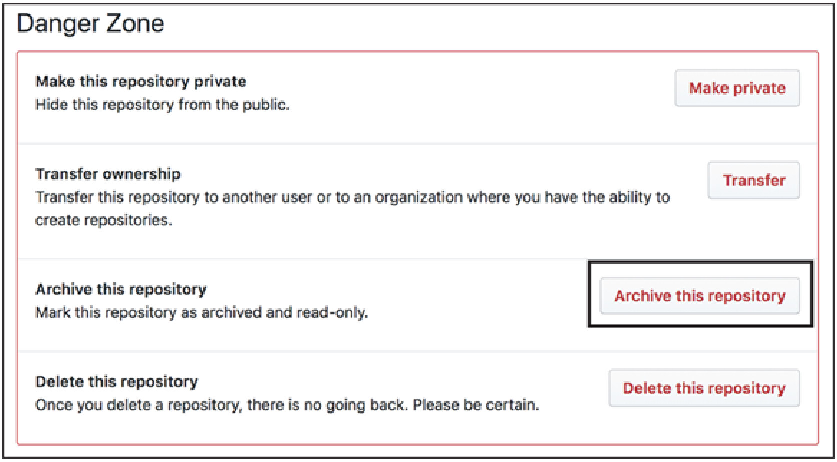
To archive a repository, go to the repository settings page:

1. **Scroll all the way down to the Danger Zone and click Archive this repository, as shown in Figure 10-18.**

Clicking this button displays a detailed confirmation box that describes what will happen if you archive the repository as shown in Figure 10-19.

2. **Type the name of the repository and click I understand the consequences, archive this repository to archive it.**

   Your repository is archived.



Danger Zone

Make this repository private
Hide this repository from the public.

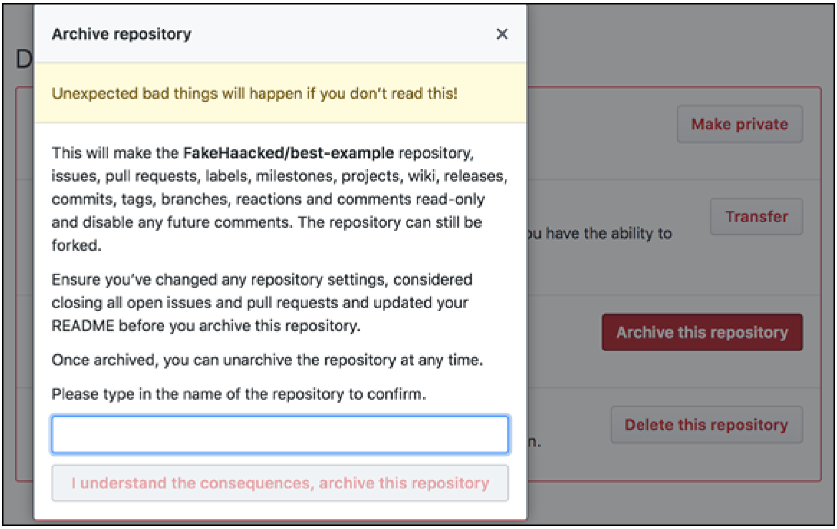
Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories.

Archive this repository
Mark this repository as archived and read-only.

Delete this repository
Once you delete a repository, there is no going back. Please be certain.

Make private
Transfer
Archive this repository
Delete this repository

FIGURE 10-18: The button to archive a repository.



Archive repository

Unexpected bad things will happen if you don't read this!

This will make the **FakeHaacked/best-example** repository, issues, pull requests, labels, milestones, projects, wiki, releases, commits, tags, branches, reactions and comments read-only and disable any future comments. The repository can still be forked.

Ensure you've changed any repository settings, considered closing all open issues and pull requests and updated your README before you archive this repository.

Once archived, you can unarchive the repository at any time.

Please type in the name of the repository to confirm.

×
I understand the consequences, archive this repository
D
Make private
Transfer
u have the ability to
Archive this repository
Delete this repository
n.

FIGURE 10-19: The button to archive a repository.

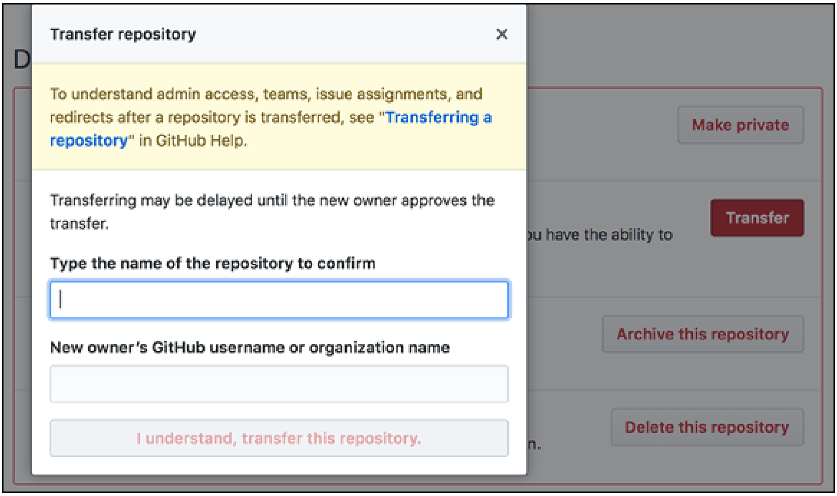
Transferring ownership

If your project is popular with a robust set of maintainers, you may want to transfer ownership to another account rather than archive the project. Transferring ownerships lets the new owner have full rights to the repository and continue to maintain it.

To transfer ownership:

1. **Go to your repository's Settings page and scroll down to the Danger Zone.**
2. **Click the Transfer button to bring up the transfer dialog box, shown in Figure 10-20.**
3. **Confirm the current repository's name and type the new owner's username or organization name.**

   The URL to the repository changes, but GitHub redirects the old URL to the new URL when anyone tries to visit the repository.


Transfer repository
×
D
Make private
Transfer
u have the ability to
Archive this repository
Delete this repository
n.
Type the name of the repository to confirm
New owner's GitHub username or organization name
I understand, transfer this repository.

To understand admin access, teams, issue assignments, and redirects after a repository is transferred, see "Transferring a repository" in GitHub Help.

Transferring may be delayed until the new owner approves the transfer.

FIGURE 10-20: The transfer repository dialog box.

Chapter 11

Inner-Source Your Code on GitHub

IN THIS CHAPTER

- **Creating a GitHub organization**
- **Setting up teams**
- **Understanding inner-source best practices**

Chapters 9 and 10 talk about the open source community and best practices on GitHub.com. But if you want to keep your code private, GitHub offers unlimited private repositories for free, personal plans to support your private software development.

In this chapter, you discover some situations where you may want to code in private. You also get the inside scoop on inner-sourcing your code.

Why Code in Private?

If we're appropriately representing the open source community, you may be inclined to do everything in the public. But what if you work at a company on proprietary code? What if you're starting a new company where you plan on selling software? Or, what if you're a student, and you're working on a group coding assignment? These are examples of when it may be appropriate to work in a private, collaborative environment, when it may be appropriate to inner-source your code.

Inner-sourcing is really just a play on words with open source. It implies that you will use the same (or similar) strategies to collaborative code writing as open source, but you will do it on private repositories.

Using GitHub Organizations

Depending on who you're working with and the scope of the project, you may want to have access to certain GitHub features. If you're working on proprietary code for a company (or code that may be a product for a future company you're hoping to start), then the investment of GitHub Teams may be suitable, giving you private repos as part of an organization. But if you're working with other students at your university on a semester-long project, then just creating a private repo may be good enough.

Creating a GitHub organization

GitHub *organizations* allow you to give a group of users access to a set of repositories all at once. With features such as *teams,* you can also have subgroups of users that have different access rights to different repositories. This setup also makes communicating across GitHub easier because you can tag an organization (or team) instead of an individual person. If you're running a large project that has more than one repository, organizations may be a good option for you.

You can start an organization in one of three tiers:

- **Team for Open Source:** In this tier, you get unlimited public repositories, unlimited collaborators, issues and bug tracking, and project management. Essentially, this tier allows a core group of people who are working on an open source project to work together easier. If you don't need any private repos associated with the organization, this tier is a great option.
- **Team:** In this tier, you get everything from the previous tier, plus unlimited private repositories, the ability to require all your organization members using two-factor authentication, team discussions that don't have to be associated to a specific repository, access for managing groups of people, and advanced tools and insights into all your repositories.

TIP *Two-factor authentication* is when you're required to enter in your password plus perform an additional security measure to ensure it is really you logging in to the website. Because passwords can sometimes be hacked, organizations will often require that you use a mobile app (such as Duo Mobile or Google Authenticator), text-message confirmation, or a physical YubiKey to verify that it is you.

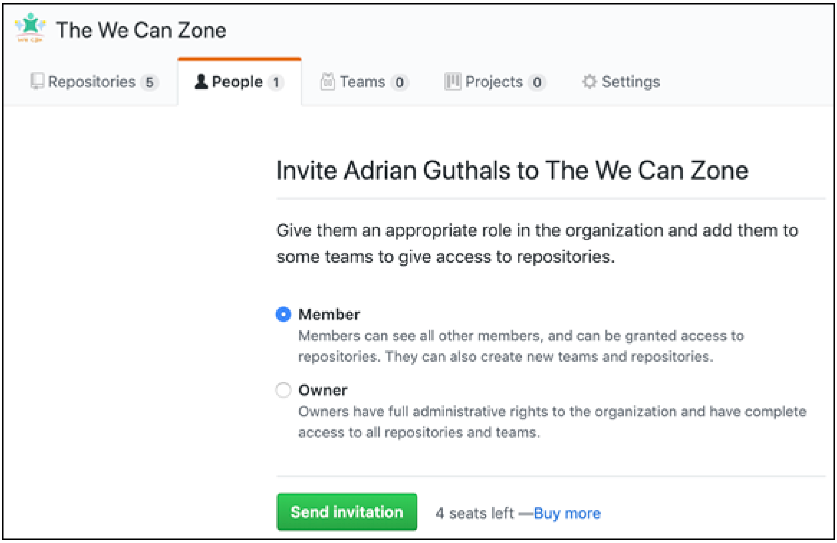
- **Enterprise:** This tier essentially gives you GitHub.com with the Team tier features, but all your data is private on a self-hosted or cloud-hosted instance of GitHub. (The data is not shared with the rest of GitHub.com users.) If you have a GitHub.com account and your company has a GitHub Enterprise instance, you will have a separate account to log in to your company's GitHub instance. This tier is most likely beyond the scope of what you'll be doing in but you may work for a company that has this.

You can find the tiers by clicking the plus symbol at the top right of GitHub.com (when you're logged in) and choosing New organization. You see a new page where you can choose the organization's name, specify a billing email address, and choose a tier.

Don't worry: If you choose the free tier, you aren't billed anything; you won't even be asked for a credit card.

Inviting members to your GitHub organization

During the organization setup process, you're asked if you want to invite others to join your organization. Using their GitHub.com alias, you can search for them in the provided box and send them an invite. If you forget to invite someone or want to invite them later, you can still do so by going to the People tab on the organization's home page and clicking Invite member. You can choose the permission level for the person you invite to your organization. Figure 11-1 shows the two options: Member and Owner. You can change the permission level later, if you need to.



Invite Adrian Guthals to The We Can Zone

Give them an appropriate role in the organization and add them to some teams to give access to repositories.

● Member
Members can see all other members, and can be granted access to repositories. They can also create new teams and repositories.

○ Owner
Owners have full administrative rights to the organization and have complete access to all repositories and teams.

4 seats left —Buy more

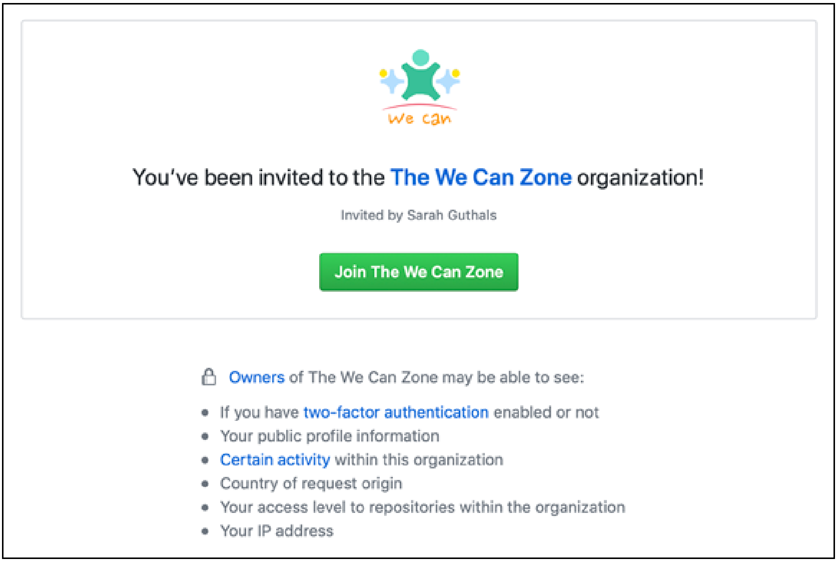
FIGURE 11-1: Inviting someone to join your organization.

Figure 11-1 also shows the number of remaining seats for your organization at the bottom of the page. If you're on the free tier, you have unlimited seats, but if you're on the paid tier, then you have only the number of seats you're paying for.

TIP

The Teams tier has a minimum of five seats at a set price. If you want to add more seats, you can, but you always start with five.

After you invite someone to your organization, they will receive an email notification with a link to accept the invitation, or they can go to the organization's GitHub home page to accept the invitation. For example, in Figure 11-1,we invited Adrian to be a part of The We Can Zone organization because he helps manage all the writing projects that we work on. When he visited `https://github.com/thewecanzone` while logged in to his GitHub account, he saw a banner and View invitation button at the top of the page. Clicking the View invitation button, he was taken to a new page, shown in Figure 11-2, where he was able to first see what kind of access the owners of the We Can Zone organization would have to his GitHub account information if he were to join.



You've been invited to the The We Can Zone organization!

Invited by Sarah Guthals

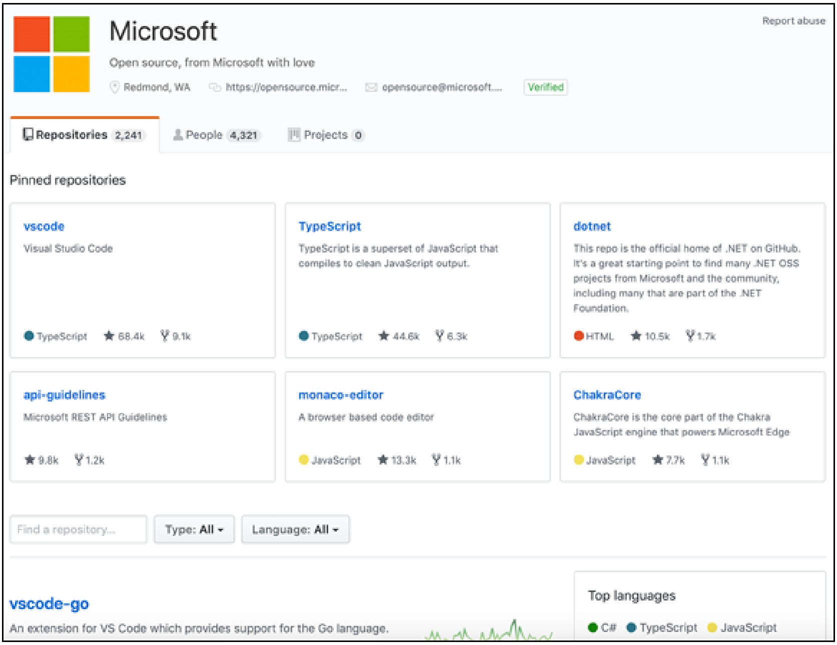
Owners of The We Can Zone may be able to see:

- If you have two-factor authentication enabled or not
- Your public profile information
- Certain activity within this organization
- Country of request origin
- Your access level to repositories within the organization
- Your IP address

FIGURE 11-2: Invitation request from the invitee's account.

Viewing repositories for your organization

Repositories is the default tab on your organization home page. This page shows you all the repositories associated with this organization. For example, if you go to the Microsoft organization home page on GitHub (`https://github.com/microsoft`), you find more than 2,000 repositories and more than 4,000 people involved in open source projects for Microsoft (see Figure 11-3).



Microsoft

Open source, from Microsoft with love

Redmond, WA · https://opensource.micr... · opensource@microsoft.... · Verified

Pinned repositories

vscode
Visual Studio Code
● TypeScript ★ 68.4k 9.1k

TypeScript
TypeScript is a superset of JavaScript that compiles to clean JavaScript output.
● TypeScript ★ 44.6k 6.3k

dotnet
This repo is the official home of .NET on GitHub. It's a great starting point to find many .NET OSS projects from Microsoft and the community, including many that are part of the .NET Foundation.
● HTML ★ 10.5k 1.7k

api-guidelines
Microsoft REST API Guidelines
★ 9.8k 1.2k

monaco-editor
A browser based code editor
● JavaScript ★ 13.3k 1.1k

ChakraCore
ChakraCore is the core part of the Chakra JavaScript engine that powers Microsoft Edge
● JavaScript ★ 7.7k 1.1k

vscode-go
An extension for VS Code which provides support for the Go language.

Top languages
● C# ● TypeScript ● JavaScript

FIGURE 11-3: Repositories associated with the Microsoft organization.

Pinned repositories appear at the top of the Repositories page. Pinned repositories are the ones that the Microsoft organization owners think are the most relevant to folks interested in what Microsoft is doing in the open source space. For example, the VS Code repository has more than 68,000 stars and more than 9,000 forks. As one of the most popular editors and most popular open source projects, Microsoft wants to make sure this repository is front and center for visitors to their open source organization home page.

Managing members of your organization

You will always have at least one member of your organization — you! But this section is more interesting if you have more than one member, so if you haven't invited other members yet, go to the earlier section "Inviting members to your GitHub organization" and invite someone else.

To see all your organization's member, from your organization home page, click the People tab. You should see all the members of your organization on this tab, as shown in Figure 11-4.

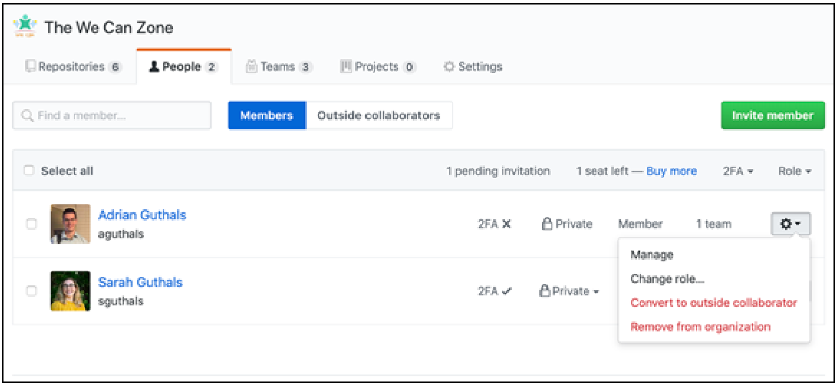

The We Can Zone
Repositories 6
People 2
Teams 3
Projects 0
Settings
Find a member…
Members
Outside collaborators
Invite member
Select all
1 pending invitation
1 seat left — Buy more
2FA
Role
Adrian Guthals
aguthals
2FA ✕
Private
Member
1 team
Manage
Change role…
Convert to outside collaborator
Remove from organization
Sarah Guthals
sguthals
2FA ✓
Private

FIGURE 11-4: A list of members of an organization.

TIP

On the right of each member is a small cog drop-down menu. This menu gives you quick options for managing your organization members. You can also get all these options and more information about a specific member by choosing Manage from that drop-down menu. You then see an overview of an organization member, as shown in Figure 11-5.

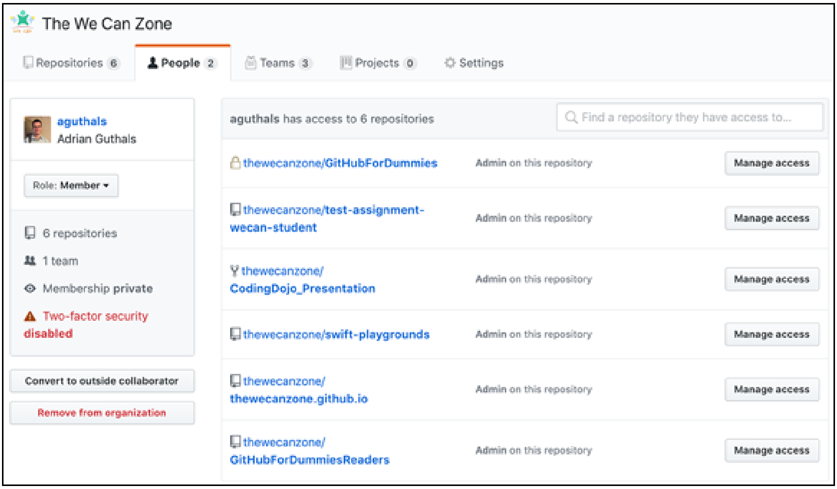

The We Can Zone
Repositories 6
People 2
Teams 3
Projects 0
Settings
aguthals
Adrian Guthals
Role: Member
6 repositories
1 team
Membership private
Two-factor security disabled
Convert to outside collaborator
Remove from organization
aguthals has access to 6 repositories
Find a repository they have access to…
thewecanzone/GitHubForDummies
Admin on this repository
Manage access
thewecanzone/test-assignment-wecan-student
Admin on this repository
Manage access
thewecanzone/CodingDojo_Presentation
Admin on this repository
Manage access
thewecanzone/swift-playgrounds
Admin on this repository
Manage access
thewecanzone/thewecanzone.github.io
Admin on this repository
Manage access
thewecanzone/GitHubForDummiesReaders
Admin on this repository
Manage access

FIGURE 11-5: An overview of a member of an organization.

Figure 11-5 gives you the following information about the person:

- **Role:** The person's role in the organization with the ability to change it to a different role from this page.
- **Repository access:** The number of repositories this person has access to within the organization, as well as a list of all the repositories and what permissions they have for each one. Each repository has a button that allows you to quickly navigate to the settings for that person for that repository.
- **Number of teams:** The number of teams the person is a part of within the organization.
- **Activity:** Information on whether the person is choosing to share their activity on projects within this organization on their public profile.
- **Two-factor authentication:** Whether this person has two-factor authentication enabled for their account. Two-factor authentication can be a requirement of your organization, which you can change in the settings for your organization. (See the upcoming section "Setting organization settings.")
- **Convert to outside collaborator:** A button to convert someone to an outside collaborator. This feature is useful for short-term or very scoped projects. Instead of having a person be a part of the entire organization, you can make them a part of a single team that has access to certain repositories, making their privileges easier to manage.
- **Remove from organization:** As straightforward as it sounds. This setting removes the person from the organization.

Creating teams within your organization

As your organization begins to grow, it may make sense for you to create teams within your organization. The benefit of teams is that you can quickly give access to a repository to an entire team, without having to remember every single person that is on that team. To create a team, click the Team tab on your organization home page and click New team. A new page appears where you can choose a team name, add a description, choose a parent team (if you've created other teams already), and set the team's visibility within the organization.

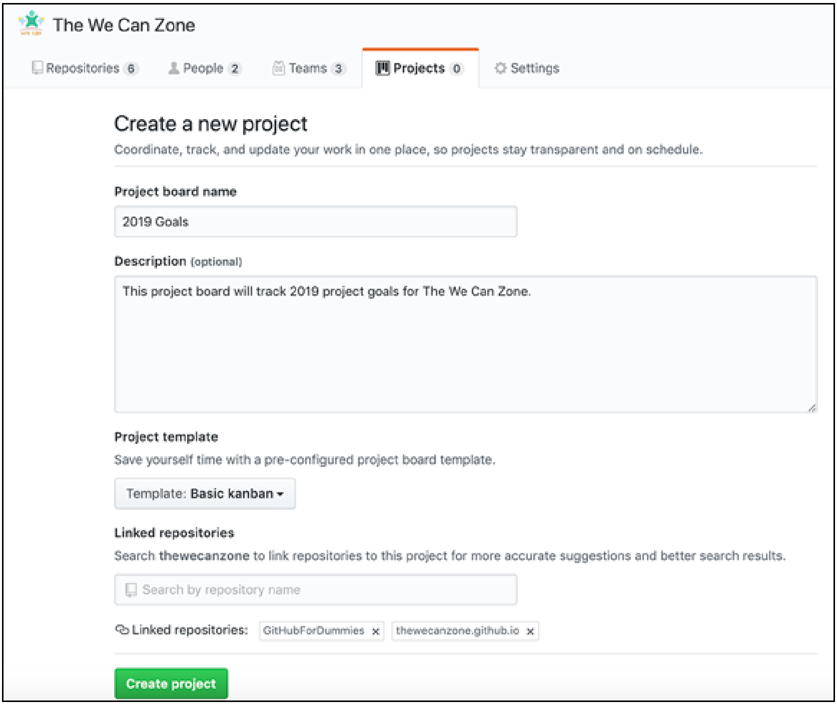
REMEMBER Choosing a team name is an important part to consider carefully. The team name will be how folks within your organization can tag everyone in the team all at once. For example, if you have a security-vulnerabilities team that manages all security vulnerabilities for your website “ and a security bug is found, you can tag the security-vulnerabilities team, and each person on that team will get a notification. This will help make sure that you get the fastest response time, accounting for different time zones, working hours, shifts, and schedules. You want the name to be representative of the team because if you had named the team something like powerpuffgirls, it would be pretty confusing to see an issue comment that says

```
@powerpuffgirls, please review this security vulnerability
   asap
```

creating Teams has a lot more benefits than the ability to mention them all using one alias. If you’re interested, read the section “Making the Most of Your Teams,” later in this chapter.

Using project boards within your organization

Chapters 3 and 4 cover how to create project boards and use them when they’re associated with a specific repository. By creating an organization, you have unlocked the ability to have project boards that link multiple repositories together. When you click the Projects tab and click the New project button, to the Create a New Project page appears, as shown in Figure 11-6.



Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name

2019 Goals

Description (optional)

This project board will track 2019 project goals for The We Can Zone.

Project template

Save yourself time with a pre-configured project board template.

Template: Basic kanban

Linked repositories

Search thewecanzone to link repositories to this project for more accurate suggestions and better search results.

Search by repository name

Linked repositories: GitHubForDummies × thewecanzone.github.io ×

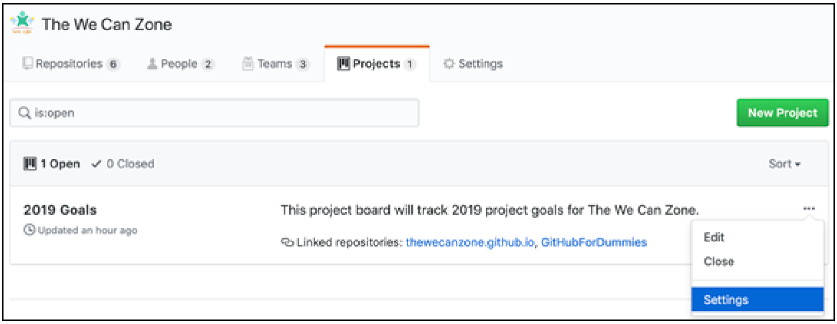
FIGURE 11-6: Creating a project board linked with multiple repositories.

Everything is the same about creating the project board as it is when you create one for a specific repository (see Chapter 2), except an option at the very bottom, right before you click Create project: a section where you can choose which repositories you want to link to this project board.

By linking multiple repositories to one project board, any rules you set for specific columns (for example, all new issues that get opened in a repository should go in a To do column) will apply to all the repositories that are linked to the project board. Linking multiple repositories to one project board can help make triaging easier for project managers, and give organization leaders a broader picture of where the entire organization is, without having to go to each repository to check in.

TIP

If you ever need to link more repositories to a project board after you create it, or change the visibility of the project or permissions for organization members or specific teams, you can click the three dot menu and navigate to the Settings page for the specific project board, as shown in Figure 11-7.



2019 Goals

Updated an hour ago

This project board will track 2019 project goals for The We Can Zone.

Linked repositories: thewecanzone.github.io, GitHubForDummies

FIGURE 11-7: Settings for organization project boards.

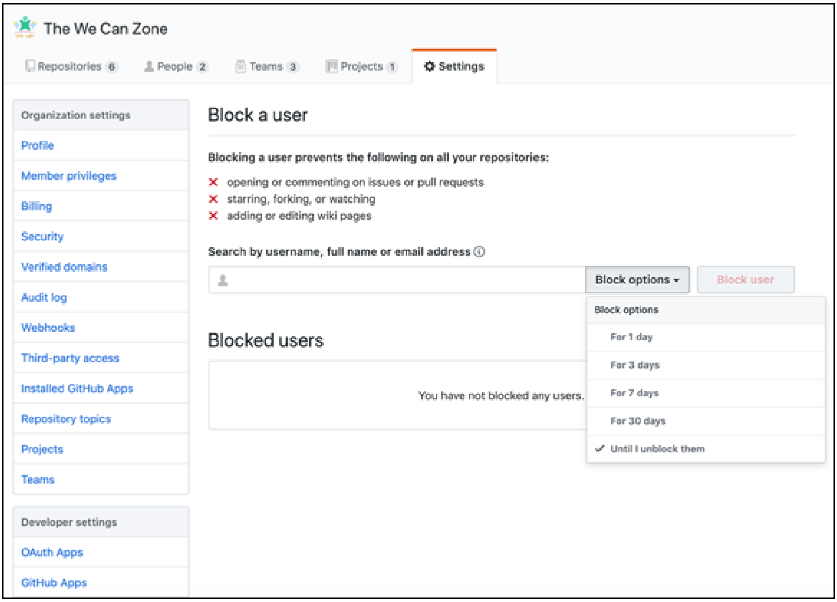
Setting organization settings

Organizations have a few more settings than typical individual GitHub accounts. On the right-most Settings tab on the organization home page a Settings page that is similar to the one we describe in Chapter 1, but with some key differences:

- **Profile** is where you can change the organization's name, avatar, and primary contact email address or even delete the organization. You can also choose to join the GitHub Developers Program (you can read about at `https://developer.github.com/program`).
- **Member privileges** is where you can set all the default permissions for every member in your organization. We recommend changing the default of new members from Admin, unless you really want every new member to have admin privileges over every repository within the organization.
- **Billing** is where you can upgrade or downgrade your organization account. You can add, change, and remove payment methods, add marketplace apps, and add seats within your organization. You can also add billing managers to your organization; this can be really useful for folks within your company who need to have access to billing information, but may not be savvy or interested in the code aspect.
- **Security** is the area where you can require that all members of

your organization have two-factor authentication setup for their GitHub.com account.

- TIP **Verified domains** is where you can verify a domain that you own so that you can verify your organization’s identity on GitHub.
- **Audit log** is a log of all activity done to the organization (not the individual repositories part of the organization).
- **Webhooks** notifies third-party services of events happening within your organization.
- **Third-party access** is all third-party applications that you have given access to your repositories.
- **Installed GitHub apps** is all the GitHub apps that you have installed within your organization.
- **Repository topics** is one place to view and modify the topics for all of your repositories.
- **Projects** contains overall settings for project boards within your organization.
- **Teams** is where you can enable team discussions, more information about this is in the section “Making the Most of Your Teams” later in this chapter.
- **Developer Settings** has two settings — OAuth Apps and GitHub Apps — and a place to specify management of the organization (under GitHub Apps).
- **Moderation** is where you can block users from your organization or enact temporary (24-hour) limitations on what activity can happen on any repository within the organization. Figure 11-8 shows that you can block users for short periods of time or forever. These moderation settings are often most useful in open source project organizations because each member is less likely to have a common driving force, such as a paycheck, to behave with respect. However, moderation settings can also be useful with inner-source projects if things simply start to get heated within the organization.


The We Can Zone
Repositories 6
People 2
Teams 3
Projects 1
Settings
Organization settings
Profile
Member privileges
Billing
Security
Verified domains
Audit log
Webhooks
Third-party access
Installed GitHub Apps
Repository topics
Projects
Teams
Developer settings
OAuth Apps
GitHub Apps
Block options
Block user
Block options
For 1 day
For 3 days
For 7 days
For 30 days
✓ Until I unblock them

Block a user

Blocking a user prevents the following on all your repositories:

- ✗ opening or commenting on issues or pull requests
- ✗ starring, forking, or watching
- ✗ adding or editing wiki pages

Search by username, full name or email address

Blocked users

You have not blocked any users.

FIGURE 11-8: Options for blocking an individual.

Making the Most of Your Teams

Having an organization with members grouped into various teams can be useful simply for understanding who is doing what, but GitHub provides you with tools that make your workflow even more effective when you group organization members into teams.

Creating parent/child teams

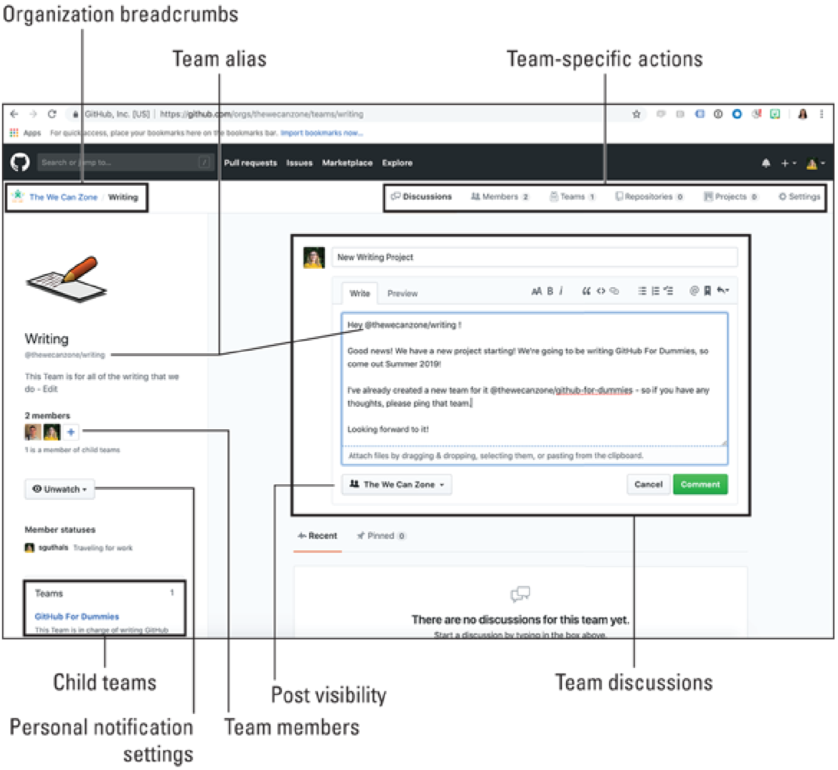
You can add a team as a parent team to a new team that you're creating (see the section "Creating teams within your organization," earlier in this chapter). Essentially, teams can have *hierarchies.* The permissions of the parent teams are passed down to all child teams, but not vice versa. Hierarchies can become extremely useful when you want to ensure that everyone within an organization has access to exactly what they need and no more, no less.

As an example, you may have a team named Employees at the top of your hierarchy. As part of that team, you may have three other teams: Human Resources, Marketing, and Engineering. Under the Human Resources team, you may have some private project boards or private repositories that only the Human Resources team members should be able to see (such as employee personal information). Under the

Marketing team, you may have customer data that shouldn't be shared publicly within the organization (such as billing account information). By assigning all the employees to respective child teams within the organization, you can ensure that everyone has access to the employee handbook, while only the folks in Human Resources have access to Social Security numbers.

Discussing teams

GitHub offers the feature of team discussions when you have an organization with teams. Figure 11-9 shows this team discussion home page, which you can find at the URL `https://github.com/orgs/ORGNAME/teams/TEAMNAME` where ORGNAME and TEAMNAME are replaced with the actual names of the organization and team, respectively.


Organization breadcrumbs
Team alias
Team-specific actions
GitHub, Inc. [US] | https://github.com/orgs/thewecanzone/teams/writing
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Search or jump to...
Pull requests
Issues
Marketplace
Explore
The We Can Zone / Writing
Discussions
Members 2
Teams 1
Repositories 0
Projects 0
Settings
Writing
@thewecanzone/writing
This Team is for all of the writing that we do - Edit
2 members
1 is a member of child teams
Unwatch
Member statuses
sguthals Traveling for work
Teams 1
GitHub For Dummies
This Team is in charge of writing GitHub
Write
Preview
The We Can Zone
Cancel
Comment
Recent
Pinned 0
There are no discussions for this team yet.
Start a discussion by typing in the box above.
Child teams
Personal notification settings
Team members
Post visibility
Team discussions

New Writing Project

Hey @thewecanzone/writing !

Good news! We have a new project starting! We're going to be writing GitHub For Dummies, so come out Summer 2019!

I've already created a new team for it @thewecanzone/github-for-dummies - so if you have any thoughts, please ping that team.

Looking forward to it!

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

FIGURE 11-9: Options for blocking an individual.

From the team discussion home page, you find

- **Organization breadcrumbs:** On the top left, the hierarchy of this team, all the way up to the organization in a clickable breadcrumb-like fashion for easy navigation.
- **Team-specific actions:** On the top right, similar menu items that are found on repositories and the organization itself, but all will be specific to this team.
- **Team discussion:** In the center right, a place to add new posts to the team discussion page. Below this area, you can find previous posts, and you can even pin specific posts to stay at the top (useful for important announcements or onboarding information).
- **Team alias:** The team alias appears below the team avatar and name on the left hand column. You can use this alias to notify everyone in the team about something that may be of interest to them, as seen in the team post.
- **Team members:** A list of all team members, a count of how many also belong to child teams, and a quick button for adding additional members to the team.
- **Personal notification settings:** On the left hand column, a place to quickly change your notification settings for this team.
- **Child teams:** A list of all child teams to this team that are linked for quick navigation, located on the bottom of the left column.

Assigning CODEOWNERS

A really neat feature that GitHub has that is enhanced with the use of teams is the CODEOWNERS feature. CODEOWNERS is a way to specify certain people who may have the most experience and knowledge on a piece of code. They're the people that, when you're looking for someone to review your code, you most likely want them to take a look. They probably have the most history with the code or are the most expert in that domain. By creating teams within your organization, you can assign multiple people to be CODEOWNERS of certain code, without having to assign each person or keep the list up to date.

To get CODEOWNERS working on one of your repositories, first make sure that you have a team setup that includes members of the organization that should be held responsible for ensuring all code that gets added to that repository is correct. In our example, I have a team called GitHub For Dummies which includes us two authors. On that team page, make sure that you've linked the repository that you want this team to be CODEOWNERS of and make sure that this team has at least Write access to that repository. Figure 11-10 shows that our team has two repositories that we have Write access to: the public one that you can access as a part of this book, and a private one that we're using to track the progress of writing this book. We're going to use CODEOWNERS for the public repository.

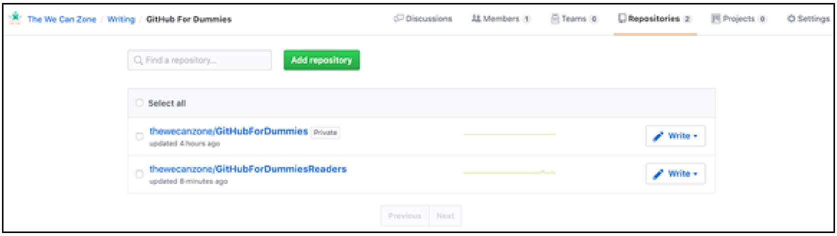

The We Can Zone / Writing / GitHub For Dummies
Discussions
Members 1
Teams 0
Repositories 2
Projects 0
Settings
Find a repository...
Add repository
Select all
thewecanzone/GitHubForDummies
Private
updated 4 hours ago
Write
thewecanzone/GitHubForDummiesReaders
updated 8 minutes ago
Write
Previous
Next

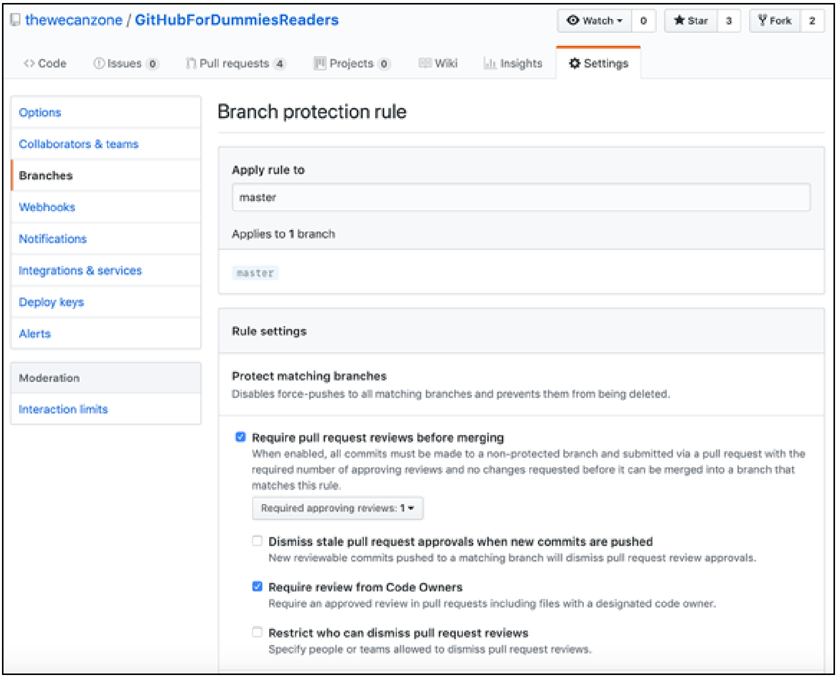
FIGURE 11-10: Repository access for a specific team.

WARNING If you give the team only Read access to the repository, then they can't approve or merge a pull request because that is technically writing to the repository. CODEOWNERs (whether individuals or teams) must have either Write or Admin access to be effective.

In the repository where you want the CODEOWNERS to be automatically added as a reviewer for all pull requests, add a file on the master branch `.github/CODEOWNERS` with the following code:

```
# These owners will be the default owners for everything in
# the repo. Unless a later match takes precedence,
# @thewecanzone/github-for-dummies will be requested for
# review when someone opens a pull request.
*       @thewecanzone/github-for-dummies
```

Make sure under the Settings for that repository, you've added a branch protection to ensure that every time someone tries to make changes to the master branch using a pull request, they have to get their code reviewed and approved by at least one person and that each pull request requires a CODEOWNER to review and approve it. You can set up these requirements in the Branches area of the Settings for the repository, as shown in Figure 11-11.


thewecanzone / GitHubForDummiesReaders
Watch 0
Star 3
Fork 2
Code
Issues 0
Pull requests 4
Projects 0
Wiki
Insights
Settings
Options
Collaborators & teams
Branches
Webhooks
Notifications
Integrations & services
Deploy keys
Alerts
Moderation
Interaction limits

Branch protection rule

Apply rule to

master

Applies to 1 branch

`master`

Rule settings

Protect matching branches
Disables force-pushes to all matching branches and prevents them from being deleted.

☑ Require pull request reviews before merging
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

Required approving reviews: 1

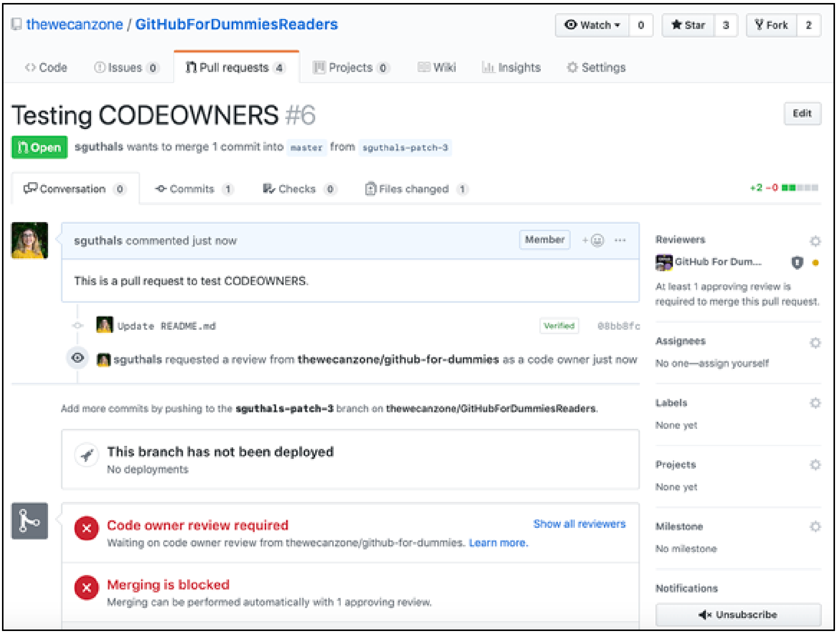
☐ Dismiss stale pull request approvals when new commits are pushed
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☑ Require review from Code Owners
Require an approved review in pull requests including files with a designated code owner.

☐ Restrict who can dismiss pull request reviews
Specify people or teams allowed to dismiss pull request reviews.

FIGURE 11-11: Requiring at least one approving review from a CODEOWNER.

Now, when you try to add code by opening a pull request, the team `@thewecanzone/github-for-dummies` will be automatically added to every single pull request, as you can see in Figure 11-12.


thewecanzone / GitHubForDummiesReaders

Watch 0 Star 3 Fork 2

Code Issues 0 Pull requests 4 Projects 0 Wiki Insights Settings

Testing CODEOWNERS #6

Edit

Open sguthals wants to merge 1 commit into `master` from `sguthals-patch-3`

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 −0

sguthals commented just now Member

This is a pull request to test CODEOWNERS.

Update README.md Verified 08bb8fc

sguthals requested a review from **thewecanzone/github-for-dummies** as a code owner just now

Add more commits by pushing to the **sguthals-patch-3** branch on **thewecanzone/GitHubForDummiesReaders**.

This branch has not been deployed
No deployments

Code owner review required Show all reviewers
Waiting on code owner review from thewecanzone/github-for-dummies. Learn more.

Merging is blocked
Merging can be performed automatically with 1 approving review.

Reviewers
GitHub For Dum...
At least 1 approving review is required to merge this pull request.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

FIGURE 11-12: CODEOWNERS automatically added to a pull request.

TIP

You can also add specific people or more teams to the CODEOWNERS file by appending them to the same line. And you can even specify what type of files you want to assign to which Code Owners. For a detailed explanation on all of the nuances, you can visit `https://help.github.com/articles/about-code-owners`.

Best Practices for Inner-Sourcing

In Chapters 9 and 10, we describe best practices for both contributing and creating your own open source project. Here is the secret: Those best practices all apply to inner-sourcing as well. Even if you're working on your code in private on a small team, it is still important to document, follow style guidelines, communicate effectively through issues and pull requests, provide applicable feedback through thorough code reviews, and be a positive team member.

REMEMBER When in doubt, pretend the world can see you.

In addition to following best practices for open source on your private projects, you can also leverage additional GitHub resources because your projects are within a private organization.

Repository insights

When you're a part of a company, evaluating an engineer on their contributions to the team can get tricky. Either as an individual trying to make sure you're making good progress or as a manager trying to write end-of-year reviews and performance ratings, you want to avoid using information to qualify someone's overall contributions and performance.

That being said, you can get some information to help guide conversations to more effectively evaluate yourself or another individual. On a repository, you can click the Insights tab at the top right. You see a list of insights about your repository and the people that contribute to it:

- **Pulse** is a snapshot of your repository for a specific time period (defaulted to one week). For example, in the month of January, 2019 the Microsoft/VSCode project had the following pulse summary:

  ```
  Excluding merges, 59 authors have pushed 1,028
     commits to master and 1,074 commits to all
     branches. On master, 1,045 files have
     changed and there have been 28,273 additions
     and 21,574 deletions.
  ```

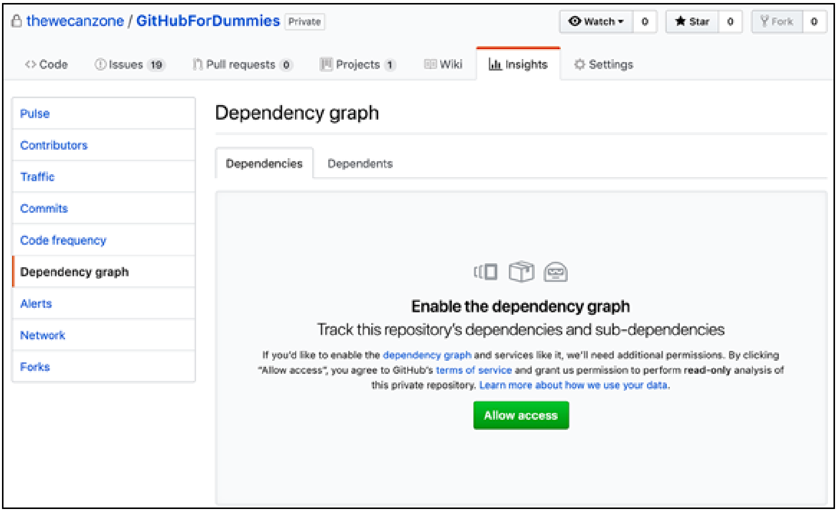
- **Contributors** shows an overall contribution graph for the repository, as well as individual contribution graphs for each contributor with a summary of the number commits, lines added, and lines removed.

  WARNING Tracking contributions to a project simply by lines of code modified or number of commits isn't an effective way to evaluate someone if nothing else is taken into account. For example, Shawn could commit every 15 minutes for fear of losing work, and Sam could always focus his energies on refactoring, making the number of lines of code he changes substantial. Comparing these two to Sandra, who had only a few commits and not as many lines of modified code but found and fixed a security vulnerability that could have taken down the application for good, isn't a fair comparison.

- **Community/Traffic:** On public repositories, the community profile on a repository helps maintainers understand where they can improve their repository to better support their community. You can find out more at `https://help.github.com/articles/about-community-profiles-for-public-repositories`. On private repositories, the Traffic section gives you insight into who is coming to this repository, where they're coming from, and what files they're interacting with.
- **Commits:** This simple commit graph can give you insights into problems your repository may have or trends with the lives of your contributors. For example, if contributions drop off every year around May, maybe a school contributes to your repository a part of the curriculum. And if you typically have 300 commits a week but suddenly you start getting only 50 commits, maybe folks are running into a problem where they can't get their code working well enough to commit.
- **Code frequency:** Similar to commits, code frequency shows the frequency at which lines of code are added and lines of code are deleted. Identifying patterns here can help you understand the health of your code as well as the profile of your contributors.
- **Dependency graph:** The dependency graph lists all dependencies (and dependents) of this repository and the version it depends on.

WARNING If you want GitHub to create a dependency graph for your private project and therefore be able to warn you if you have any dependency security vulnerabilities, you have to grant GitHub read-only access to your private repo (see Figure 11-13). (For more on dependency security vulnerabilities, read `https://github.blog/2018-07-12-security-vulnerability-alerts-for-python/` for the announcement on security vulnerability alerts for Python.)

- **Alerts:** On a private repository, you have an Alerts section. If you've enabled read-only access to GitHub either via allowing it through the dependency graph setting shown in Figure 11-13 or in the Settings tab for the repository in the Options category under Data Services, as shown in Figure 11-14, you will see security alerts here.
- **Network:** The network graph shows all the people that have branched and forked the repository and any branches of branches or forks of forks. Essentially, it shows all the possible states of your repository in the world today. Figure 11-15 shows the network graph for the GitHub package for the Atom editor.
- **Forks:** The Forks tab lists links to all forks of your repository.

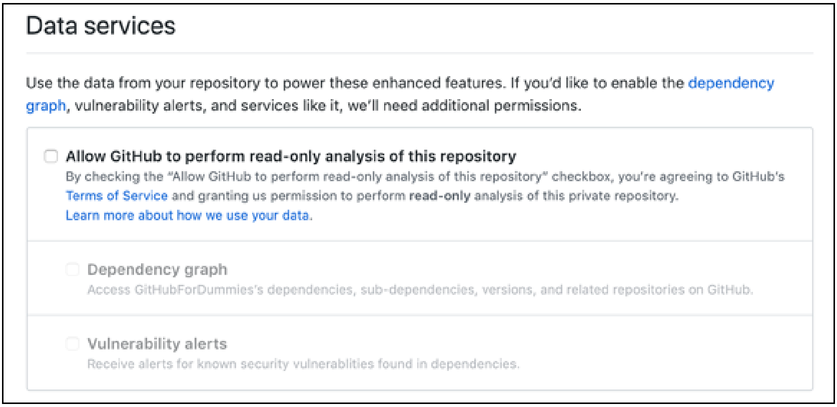

thewecanzone / GitHubForDummies
Private
Watch 0
Star 0
Fork 0
Code
Issues 19
Pull requests 0
Projects 1
Wiki
Insights
Settings
Pulse
Contributors
Traffic
Commits
Code frequency
Dependency graph
Alerts
Network
Forks
Dependency graph
Dependencies
Dependents
Allow access

Enable the dependency graph

Track this repository's dependencies and sub-dependencies

If you'd like to enable the dependency graph and services like it, we'll need additional permissions. By clicking "Allow access", you agree to GitHub's terms of service and grant us permission to perform read-only analysis of this private repository. Learn more about how we use your data.

FIGURE 11-13: GitHub must have read access to your repository to give security alerts.



Data services

Use the data from your repository to power these enhanced features. If you'd like to enable the dependency graph, vulnerability alerts, and services like it, we'll need additional permissions.

☐ **Allow GitHub to perform read-only analysis of this repository**
By checking the "Allow GitHub to perform read-only analysis of this repository" checkbox, you're agreeing to GitHub's Terms of Service and granting us permission to perform **read-only** analysis of this private repository.
Learn more about how we use your data.

☐ Dependency graph
Access GitHubForDummies's dependencies, sub-dependencies, versions, and related repositories on GitHub.

☐ Vulnerability alerts
Receive alerts for known security vulnerablities found in dependencies.

FIGURE 11-14: Allow GitHub read access to your repository for various data services.

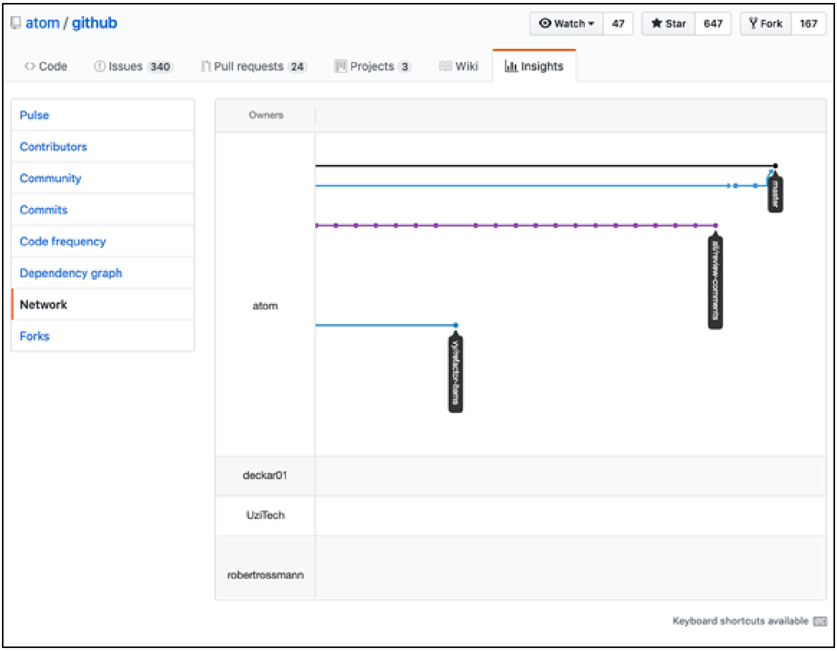

atom / github
Watch 47
Star 647
Fork 167
Code
Issues 340
Pull requests 24
Projects 3
Wiki
Insights
Pulse
Contributors
Community
Commits
Code frequency
Dependency graph
Network
Forks
Owners
atom
master
sb/review-comments
ys/refactor-items
deckar01
UziTech
robertrossmann
Keyboard shortcuts available

FIGURE 11-15: The network graph for the GitHub package for Atom open source project.

Milestones for larger projects

Project boards are an effective way to track the progress you're making through the issues and pull requests, but when you're working as part of a larger organization, you will often have larger milestones that you're trying to reach. GitHub provides support for milestones that can be linked to issues and pull requests.

To create a milestone:

1. **Go to the Issues tab of your repository and click the Milestones button.**

   If you don't have a milestone yet, you can click the big, green New milestone button on the top right or the big, green Create a Milestone button in the center. This will take you to a new page where you can add a title and optional due date and optional description.

2. **Click Create milestone.**

   You see a list of milestones.

In your list of milestones, to the right of the one you just created, is a progress bar with a status of the percentage of issues and pull requests completed, the total number of issues and pull requests still open, and the total number of issues and pull requests already closed. This information can give you a quick snapshot of how close you are (or, in most cases, aren't) to meeting your deadline.

On your Issues or Pull requests tabs, you can add the milestone to any issue or pull request, and the status icon on the milestone list updates automatically. Clicking the Milestone gives you a list of all the issues and pull requests associated with that particular milestone.

Part 5

Make GitHub Work For You

IN THIS PART …

Explore ways to keep informed of GitHub repository activity outside of GitHub.

Install popular GitHub integrations into Slack and Trello.

Set up Octobox to manage GitHub notifications.

Enhance your editor with GitHub extensions to support your collaborative workflow.

Create a probot app to personalize your GitHub.com experience.

Discover GitHub Actions to customize your GitHub.com workflow.

Chapter 12

Collaborating Outside of GitHub

IN THIS CHAPTER

- **Setting up GitHub integration in Slack**
- **Setting up GitHub integrations in Trello**
- **Installing Octobox to manage notifications**

While GitHub may be the hub of a software development project (it hosts the key deliverable, the source code), it is not the only place where collaboration occurs. Software development teams use a variety of tools to communicate and coordinate their software efforts. Many people who are not developers also work on a software project and will want to keep apprised of the progress of a project in the tools they use.

For example, a lot of day-to-day collaboration occurs in chat rooms, such as Slack. Others may use a Trello board to manage tasks for a team. Still others may use Octobox to keep on top of their GitHub notifications.

In this chapter, we explore the various integrations that bring GitHub information into other collaboration tools. This chapter is in contrast to Chapter 13 where we cover integrations that bring information from other tools into GitHub to improve the software development workflow.

Chatting It Up

For many teams, especially distributed teams, chat is a powerful way for members of the team to collaborate and coordinate their efforts. Chat in this context does not refer to sipping tea on a porch talking about how their day went. *Chat* refers to text-based tools, such as Slack, used by teams to communicate both synchronously and asynchronously.

Many teams find it helpful to have GitHub post important notifications into a chat room so teams are kept apprised of what's going on with a repository. In this section, we set up a GitHub integration with one popular chat software, Slack.

Before you install the integration, you need to be the admin of a Slack workspace. You can create a free Slack workspace at `https://slack.com`.

After you set up your Slack workspace, installing the GitHub for Slack integration requires two key steps:

1. **Install the GitHub app for Slack in the Slack workspace.**
2. **Add the Slack App for GitHub to your GitHub account.**

The following sections cover these steps in detail.

Installing the GitHub app for Slack

To install the GitHub app for Slack:

1. **Go to `https://slack.github.com` and click the Add to Slack button in the center of the browser window.**

   If you're not logged into your Slack workspace in the browser, clicking the Add to Slack button prompts you to sign into your slack workspace. Likewise, if you're not logged into GitHub, the site prompts you to log into GitHub. When you're authenticated to both, you see a Slack confirmation screen with information on what permissions the GitHub app will have to your Slack workspace. Figure 12-1 shows the confirmation screen with every section expanded (they're collapsed by default), so you can see everything the integration can do.

TIP Be sure you’re adding the GitHub app to the correct Slack workspace. If the wrong workspace is shown, you can change it by clicking the workspace drop-down list on the top right of this screen. Here, you can choose to change which Slack workplace you want to install the integration into. If the workspace isn’t listed there, you can click Sign in to another workspace.

2. **Click Continue to go to the GitHub app configuration screen.**

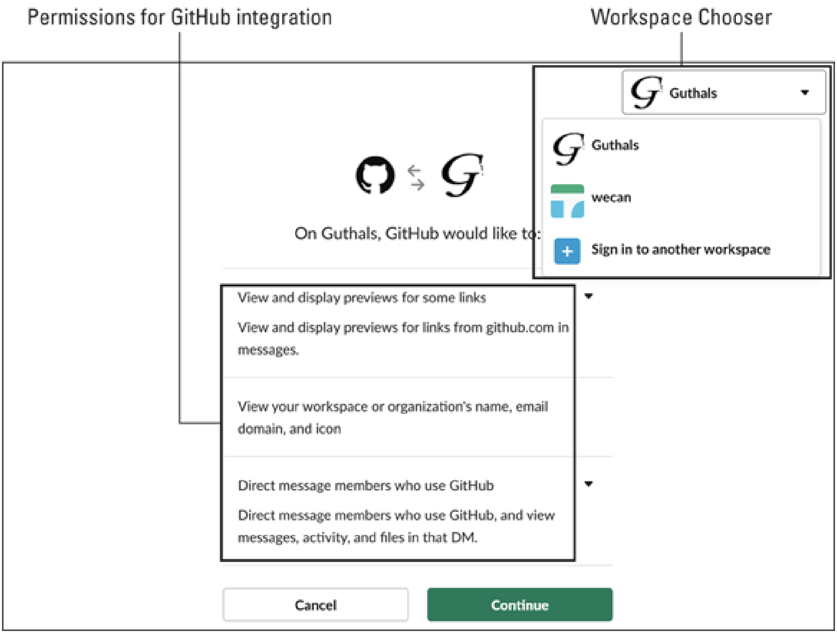
   As shown in Figure 12-2, this screen prompts you to pick which channels this app should be available to.

3. **Choose All Public Channels.**

   Any public channel in your Slack workspace can choose to make use of this app and subscribe to GitHub repository notifications.

4. **Click the Install button to complete the installation on the Slack side.**

   You’re redirected back to Slack.


Permissions for GitHub integration
Workspace Chooser
Guthals
Guthals
wecan
Sign in to another workspace
Cancel
Continue

On Guthals, GitHub would like to:

View and display previews for some links

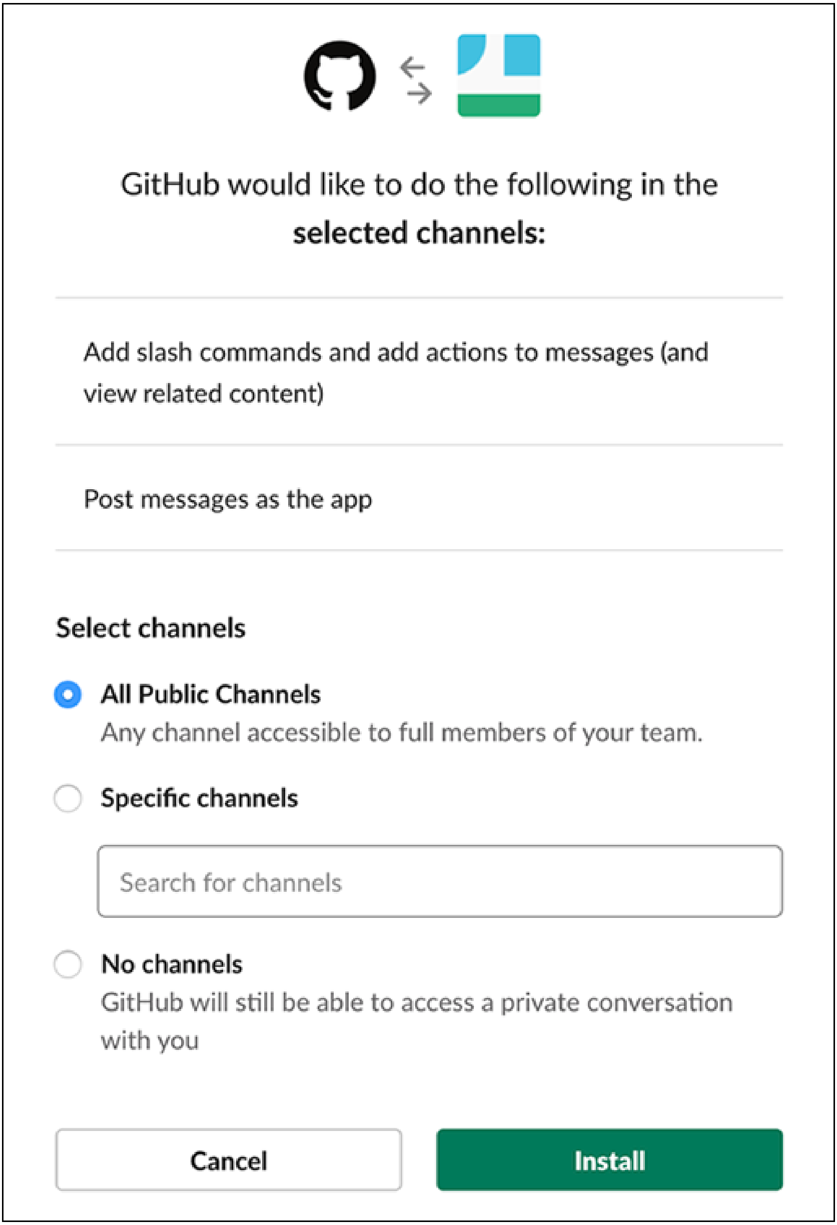
View and display previews for links from github.com in messages.

View your workspace or organization's name, email domain, and icon

Direct message members who use GitHub

Direct message members who use GitHub, and view messages, activity, and files in that DM.

FIGURE 12-1: Confirmation page for the GitHub app.



GitHub would like to do the following in the **selected channels:**

Add slash commands and add actions to messages (and view related content)

Post messages as the app

Select channels

● **All Public Channels**
Any channel accessible to full members of your team.

○ **Specific channels**

Search for channels

○ **No channels**
GitHub will still be able to access a private conversation with you

FIGURE 12-2: Choose which channels the GitHub app is enabled in.

WARNING If you have the Slack desktop application, clicking the Install button attempts to launch the application and take you to the workspace where you installed the app. If you haven't yet added the workspace to the desktop application, you'll just be in the last workspace you used. This can be a bit confusing as it may seem like the installation didn't work. Don't worry, it probably did work. Just add the workspace to your desktop application and continue.

Subscribing to a repository in a Slack channel

After you install the GitHub app for Slack, you can subscribe to notifications for a GitHub repository from within a Slack channel:

1. **Type the following command:**

   ```
   /github subscribe owner/repository
   ```

   For example, to subscribe to the repository we created for the readers of this book, `https://github.com/thewecanzone/GitHubForDummiesReaders`, you would type the following in a Slack channel:

   ```
   /github subscribe TheWecanZone/
     GitHubForDummiesReaders
   ```

   The first time you run this command, Slack prompts you to connect your Slack account to your GitHub account. as shown in Figure 12-3. This Slackbot message is visible only to you.

2. **Click the Connect GitHub Account button to start the process of connecting your account.**

   If you've already authenticated and authorized Slack on GitHub.com by either doing these steps previously or by following the steps in the section "Installing the GitHub app for Slack," earlier in this chapter, your browser will open and automatically authenticate and redirect you to Slack.

If you’re not authenticated on GitHub.com, GitHub prompts you to authenticate first and complete the rest of the steps in this section.

After authenticating, GitHub prompts you to authorize the Slack app, as shown in Figure 12-4. You won’t need to do this again when installing GitHub into other Slack workspaces.

3. **Click Authorize Slack by GitHub to continue.**

   After you authorize the app, GitHub prompts you to choose an org to install the app into or to choose your account.

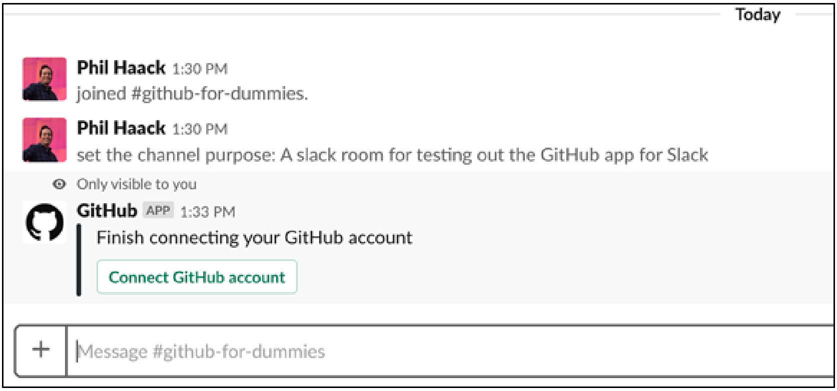
4. **Click the account name corresponding to the repository you want to subscribe to.**

   In this example, we are installing to a repository in TheWeCanZone org, so we would click that option. Clicking the account takes you to next step where you specify which repositories the app may access, as shown in Figure 12-5.

5. **Choose All repositories to enable the app for all current and future repositories or click Only select repositories and pick the specific repositories that may use the app.**

6. **After you select the set of repositories, click the Install button to complete the installation on GitHub.**

   Slack is now listed in the Applications page of your account or organization settings. You can go there to change the settings for the app or to uninstall it.


Today

Phil Haack 1:30 PM
joined #github-for-dummies.

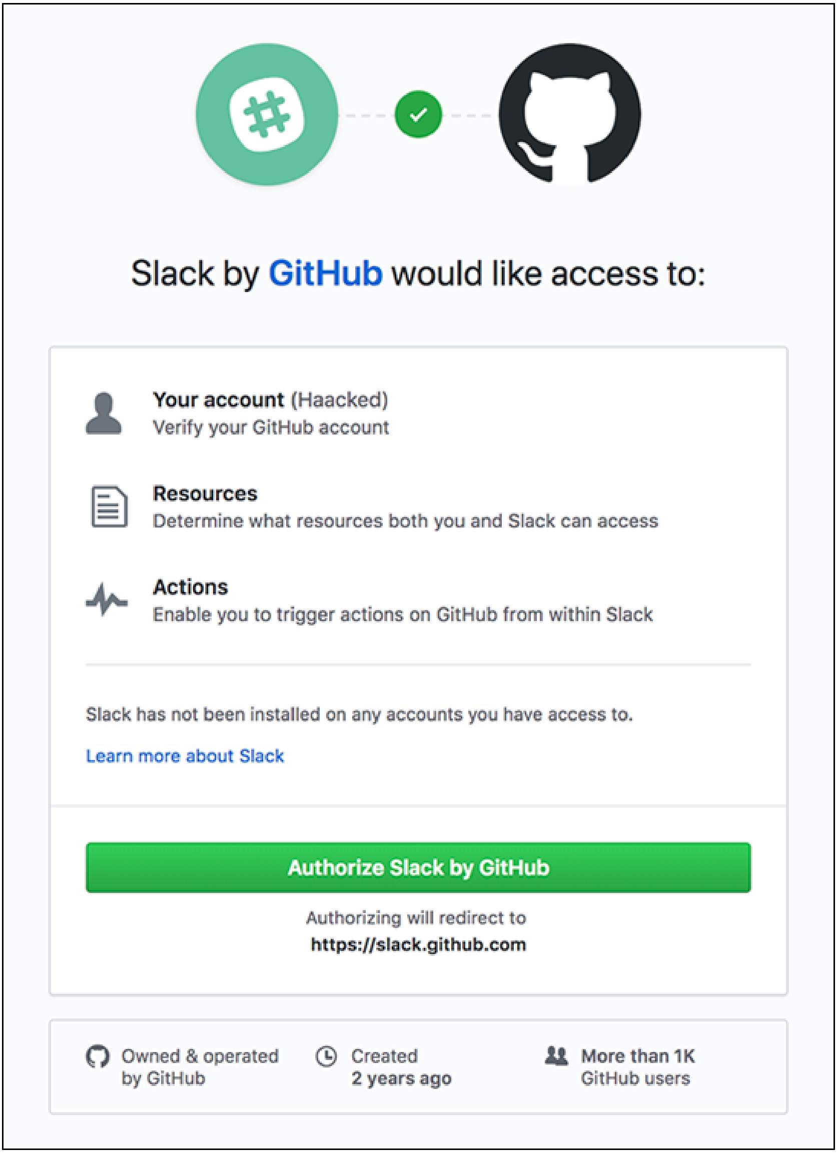
Phil Haack 1:30 PM
set the channel purpose: A slack room for testing out the GitHub app for Slack

Only visible to you

GitHub APP 1:33 PM
Finish connecting your GitHub account

Connect GitHub account
+
Message #github-for-dummies

FIGURE 12-3: Slack message with a prompt to connect your GitHub account.



Slack by GitHub would like access to:

Your account (Haacked)
Verify your GitHub account

Resources
Determine what resources both you and Slack can access

Actions
Enable you to trigger actions on GitHub from within Slack

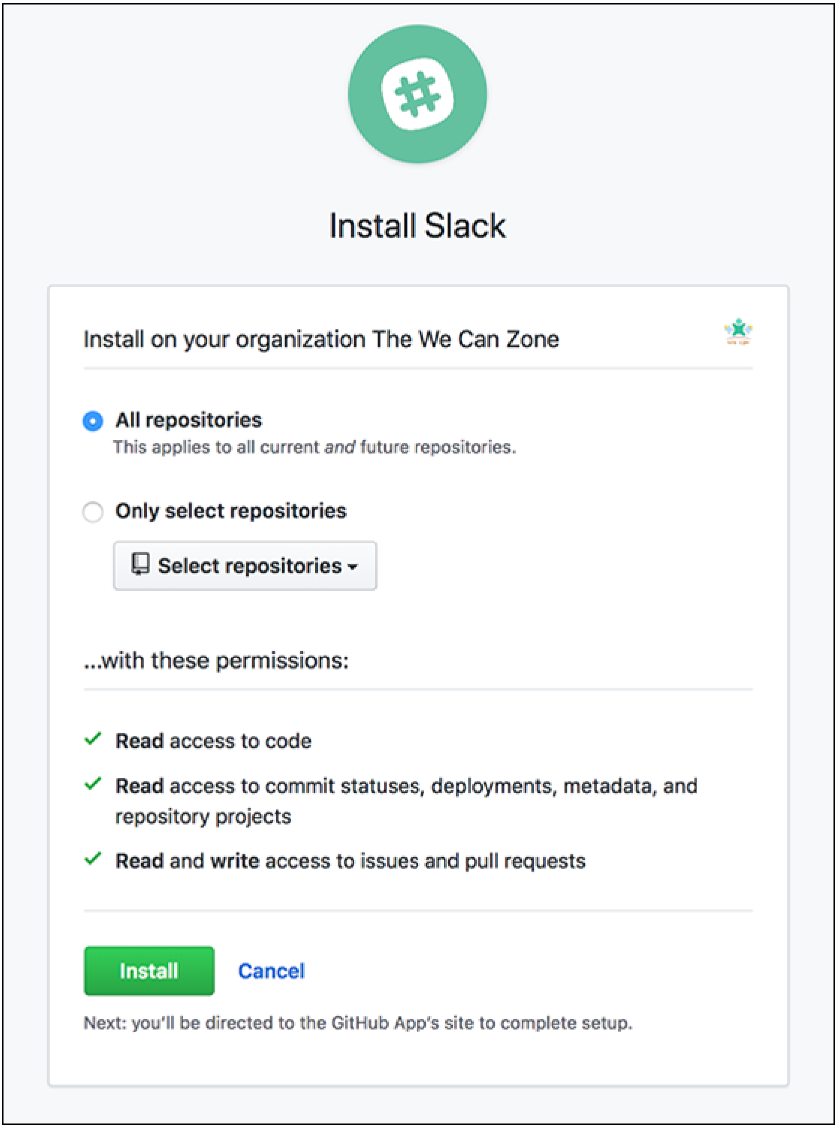
Slack has not been installed on any accounts you have access to.

Learn more about Slack

Authorizing will redirect to
https://slack.github.com

Owned & operated by GitHub
Created 2 years ago
More than 1K GitHub users

FIGURE 12-4: Authorize the Slack app for GitHub.



Install on your organization The We Can Zone

● **All repositories**
This applies to all current *and* future repositories.

○ **Only select repositories**

...with these permissions:

- ✓ Read access to code
- ✓ Read access to commit statuses, deployments, metadata, and repository projects
- ✓ Read and write access to issues and pull requests

Next: you'll be directed to the GitHub App's site to complete setup.

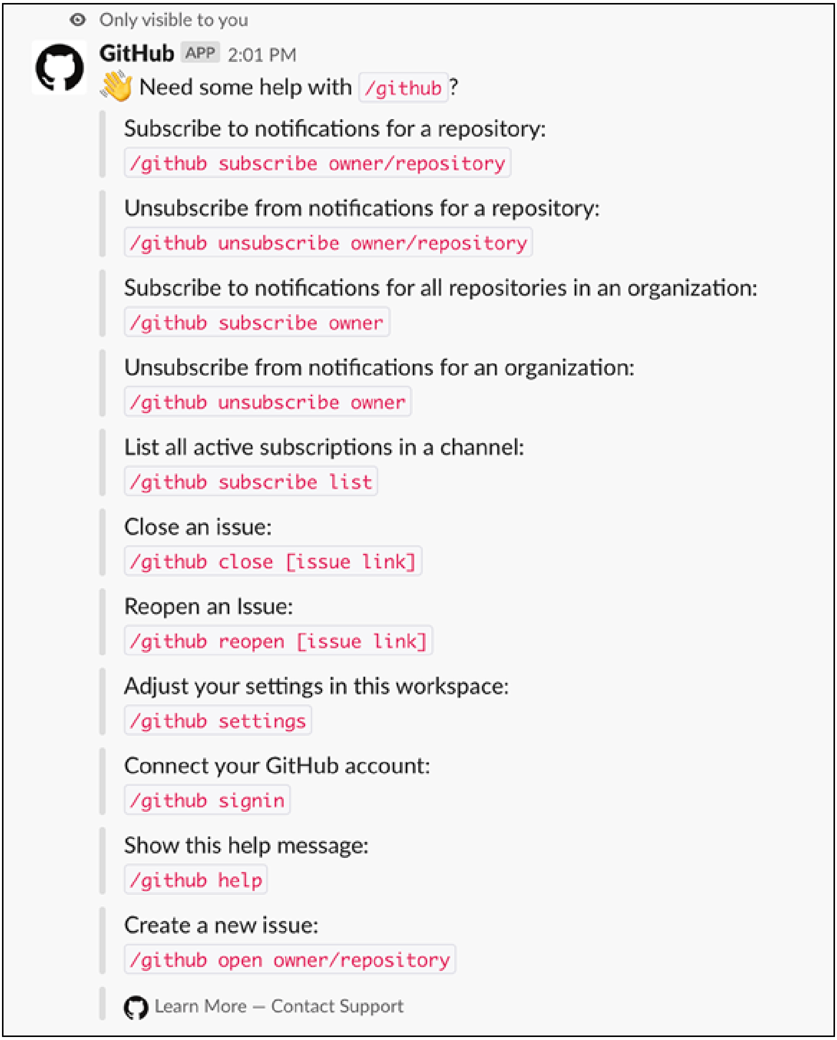
FIGURE 12-5: Configuring the Slack app on GitHub.

Trying out the GitHub Slack integration

With the installation complete, you can now subscribe to GitHub repositories in your Slack channels. To see the full list of Slack commands, type the following:

```
/github help
```

The output of this command is shown in Figure 12-6.



Only visible to you

GitHub APP 2:01 PM

👋 Need some help with `/github`?

Subscribe to notifications for a repository:
`/github subscribe owner/repository`

Unsubscribe from notifications for a repository:
`/github unsubscribe owner/repository`

Subscribe to notifications for all repositories in an organization:
`/github subscribe owner`

Unsubscribe from notifications for an organization:
`/github unsubscribe owner`

List all active subscriptions in a channel:
`/github subscribe list`

Close an issue:
`/github close [issue link]`

Reopen an Issue:
`/github reopen [issue link]`

Adjust your settings in this workspace:
`/github settings`

Connect your GitHub account:
`/github signin`

Show this help message:
`/github help`

Create a new issue:
`/github open owner/repository`

Learn More — Contact Support

FIGURE 12-6: Summary of available GitHub commands on Slack.

To test the GitHub app and open a new issue:

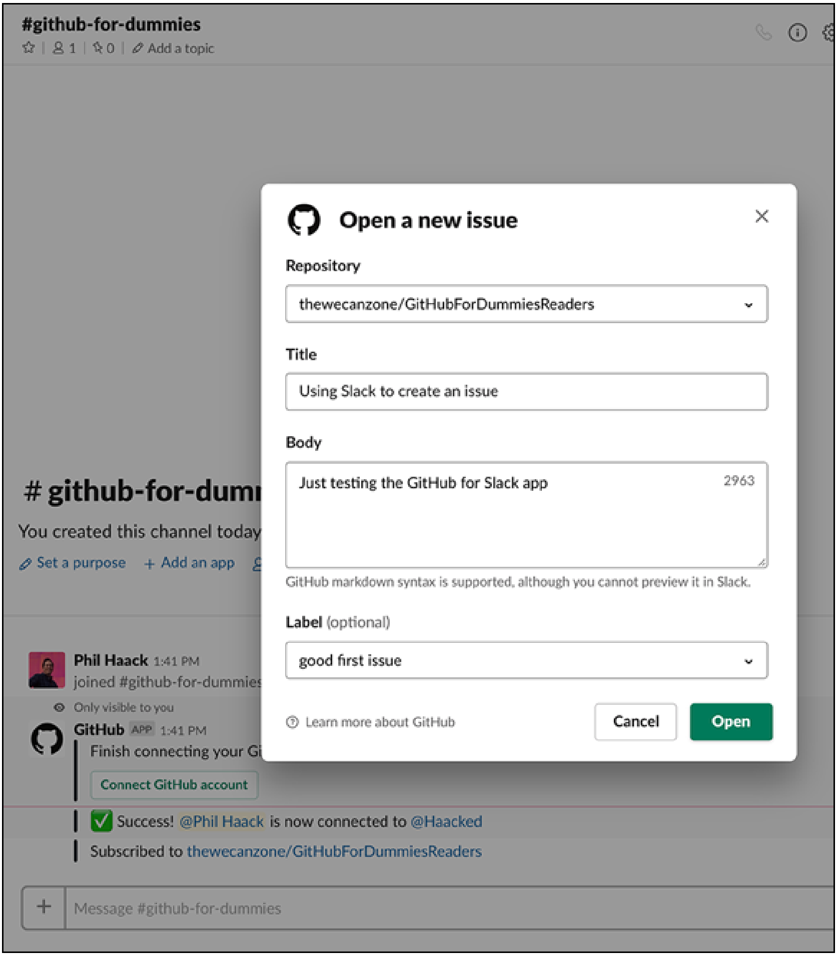
1. **Run the following command.**

```
/github open TheWeCanZone/
  GitHubForDummiesReaders
```

A Slack dialog box appears. You can use this dialog box to create a new issue, as shown in Figure 12-7.

2. **Fill in the dialog box and click the Open button.**

   Clicking the Open button creates the issue on GitHub. And because we're subscribed to that repository, we get a Slack message in the channel that the issue was created as shown in Figure 12-8.



#github-for-dummies
1 | 0 | Add a topic
Open a new issue
Cancel
Open
Learn more about GitHub
# github-for-dumm
Set a purpose
+ Add an app
Connect GitHub account
Message #github-for-dummies

Repository: thewecanzone/GitHubForDummiesReaders

Title: Using Slack to create an issue

Body: Just testing the GitHub for Slack app 2963

GitHub markdown syntax is supported, although you cannot preview it in Slack.

Label (optional): good first issue

You created this channel today

Phil Haack 1:41 PM
joined #github-for-dummies

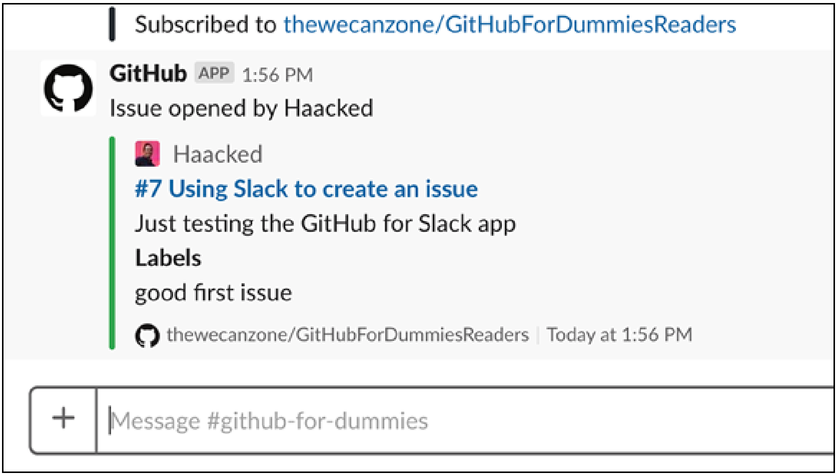
Only visible to you

GitHub APP 1:41 PM
Finish connecting your Gi

✅ Success! @Phil Haack is now connected to @Haacked

Subscribed to thewecanzone/GitHubForDummiesReaders

FIGURE 12-7: Slack dialog to create an issue on GitHub.



Subscribed to thewecanzone/GitHubForDummiesReaders

GitHub APP 1:56 PM

Issue opened by Haacked

Haacked

#7 Using Slack to create an issue

Just testing the GitHub for Slack app

Labels

good first issue

thewecanzone/GitHubForDummiesReaders | Today at 1:56 PM

FIGURE 12-8: Slack message with information about a newly created GitHub issue.

TIP

If you find a bug with the GitHub integration or have an idea for a way it could be better, good news! It's open source on, of course, GitHub! You can log issues or even contribute at `https://github.com/integrations/slack`.

The `/github subscribe` command by default subscribes a channel to notifications for the following features of a repository:

- `issues`: Opened or closed issues
- `pulls`: New or merged pull requests
- `statuses`: Statuses on pull requests
- commits: New commits on the default branch (usually master)
- `deployments`: Updated status on deployments
- `public`: A repository switching from private to public
- `releases`: Published releases

To subscribe to only a single feature, use the `/github subscribe owner/repo [feature]` command.

```
/github subscribe TheWeCanZone/GitHubForDummiesReaders reviews
```

You can remove a single feature by using the `/github unsubscribe owner/repo [feature]` command. For example, to remove commit notifications on the default branch, run the following command.

```
/github unsubscribe TheWeCanZone/GitHubForDummiesReaders
   commits
```

Additional features are disabled by default:

- `reviews`: Pull request reviews
- `comments`: New comments on issues and pull requests
- `branches`: Created or deleted branches
- `commits:all`: All commits pushed to any branch

Getting Trello and GitHub Integrated

Trello is a collaboration tool used to organize projects into boards, lists, and cards. It's inspired by the Kanban scheduling system popularized by Toyota. Kanban is Japanese for signboard. The idea is to have a board that provides a view of a project’s status and progress at a glance.

Often, a tool like Trello is combined with GitHub to manage a project. A project team may use Trello to manage the entire project, but use GitHub to host the code and assign specific code issues to developers. A card in Trello might correspond to multiple GitHub issues.

TIP GitHub Project Boards are essentially Trello with GitHub already integrated into it. However, Project Boards within GitHub aren’t always the right fit for your team. Some teams have nondeveloper folks who don’t want to have to learn GitHub and may already be using Trello. Typically, it’s best to have all your project management in one place, so if that should be outside of GitHub, in Trello for example, you can still make it a part of your developer workflow with this integration.

A GitHub integration (what Trello calls a *power-up*) for Trello connects cards to GitHub issues, pull requests, and branches. In the next section, we walk through setting up the GitHub power-up.

Installing the GitHub power-up

The following installation instructions assume that you've already signed up for `https://trello.com` and created a project board:

TIP If you've never used Trello, you can visit its guides at `https://trello.com/en-US/guide`. It is similar to GitHub Project Boards (see Chapter 3). For specific help on creating a board and cards, visit `https://trello.com/guide/create-a-board.html`.

1. **With a board open, make sure the menu is open.**

   If not, click in the top right to show the menu.

2. **Click the Power-Ups section of the menu, as shown in Figure 12-9.**

   Clicking the Power-Ups button brings up a search dialog box for power-ups.

3. **Search for GitHub to find GitHub related power-ups.**
4. **Click the Add button for the GitHub power-up to enable it.**

   TIP Just like for any application, Trello may have GitHub power-ups (extensions/integrations) that are built by GitHub and some that are built by other folks. Since GitHub's API is public, folks can often create power-ups/extensions of their own. Be sure you're always aware of the author of the power-up/extension when you're installing it. You may very well want to install from a third-party developer instead of GitHub itself because the features might be different. Regardless, you should make sure you're aware of that choice. Don't assume anything with "GitHub" in the title is made by GitHub the company.

5. **After you enable the power-up, click the gear icon in the corner to configure it.**

   You see a menu with the option to disable the power-up, or authorize it.

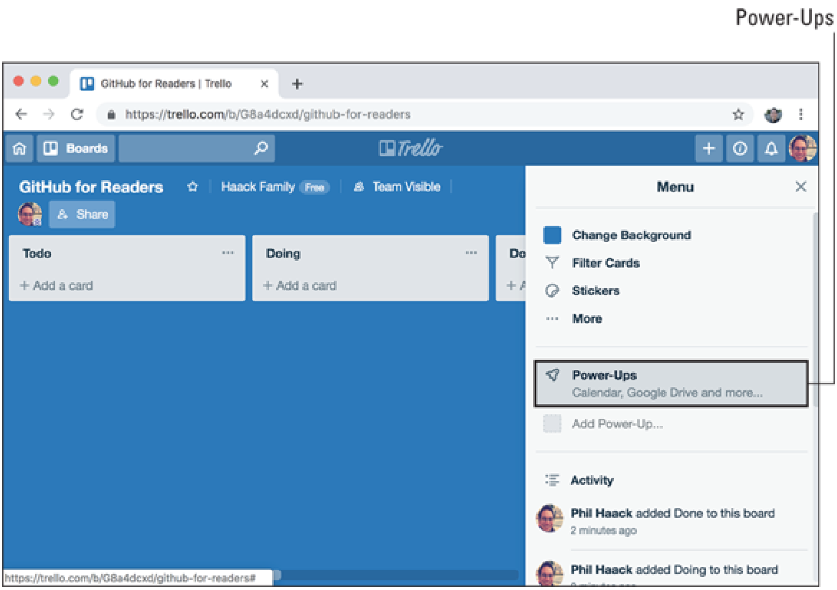
6. **Click Authorize Account.**

   An option to link your GitHub account appears.

7. **Click Link Your GitHub Account.**

   GitHub.com launches in your browser and prompts you to Authorize Trello, as shown in Figure 12-10.

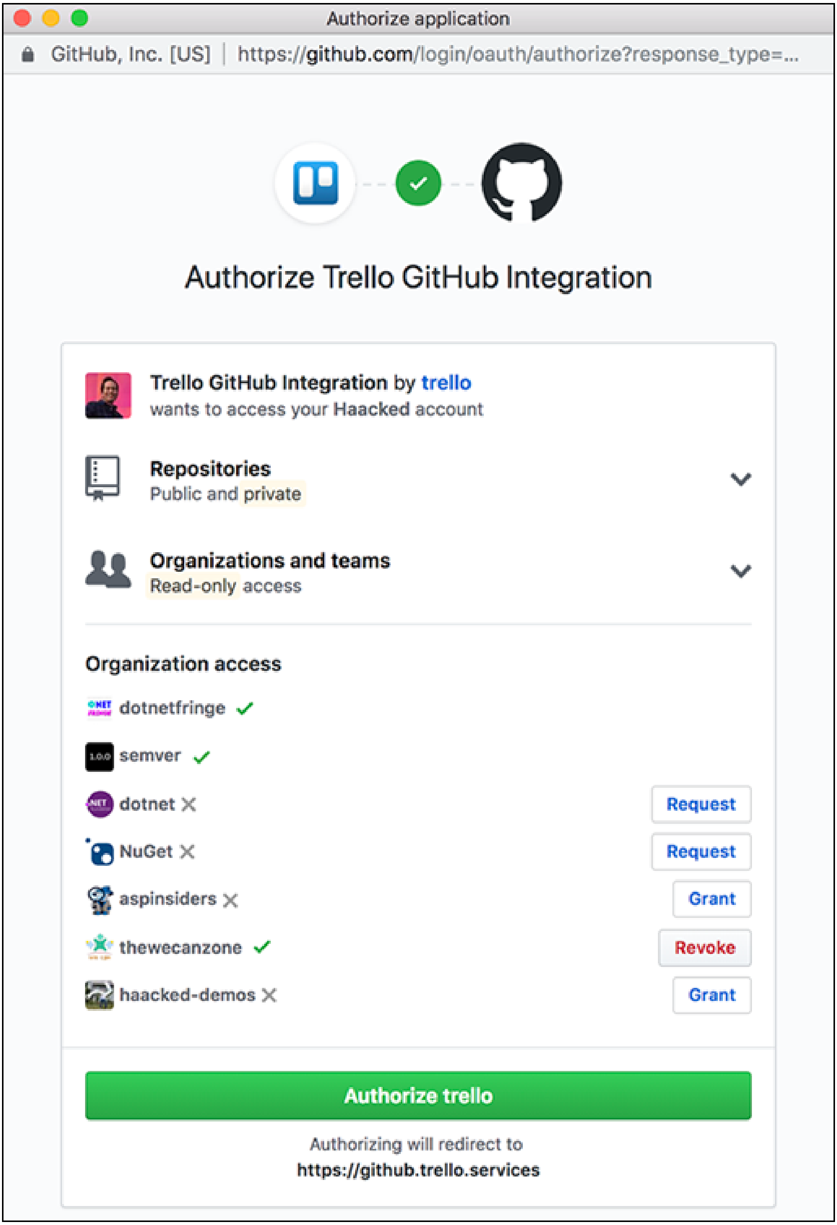
   Before clicking the Authorize trello button, make sure to click the Grant button next to any organizations that you want to connect with Trello. In my case, I'll grant Trello access to the thewecanzone organization and then click the Authorize trello button to make the power-up active.


Power-Ups
GitHub for Readers | Trello
https://trello.com/b/G8a4dcxd/github-for-readers
Boards
Trello
GitHub for Readers
Haack Family Free
Team Visible
Share
Todo
+ Add a card
Doing
+ Add a card
Menu
Change Background
Filter Cards
Stickers
More
Power-Ups
Calendar, Google Drive and more...
Add Power-Up...
Activity
https://trello.com/b/G8a4dcxd/github-for-readers#

Phil Haack added Done to this board
2 minutes ago

Phil Haack added Doing to this board

FIGURE 12-9: Power-Ups section of the menu.


Authorize application
GitHub, Inc. [US] | https://github.com/login/oauth/authorize?response_type=...
Request
Request
Grant
Revoke
Grant
Authorize trello

Authorize Trello GitHub Integration

Trello GitHub Integration by trello
wants to access your **Haacked** account

Repositories
Public and private

Organizations and teams
Read-only access

Organization access

- dotnetfringe ✓
- semver ✓
- dotnet ✗
- NuGet ✗
- aspinsiders ✗
- thewecanzone ✓
- haacked-demos ✗

Authorizing will redirect to
https://github.trello.services

FIGURE 12-10: Authorize Trello on GitHub.

Using the GitHub power-up

The GitHub power-up is accessed via the power-up button on the back of any Trello card. If you haven’t already, go ahead and create a couple of cards.

To use the GitHub power-up on your Trello board, follow these steps:

1. **Click the card to access the back of the card.**

   Figure 12-11 shows a card that we created. The GitHub power-up shows up in the bottom right corner.

2. **Click the GitHub Power-Up button.**

   Four menu options appear:

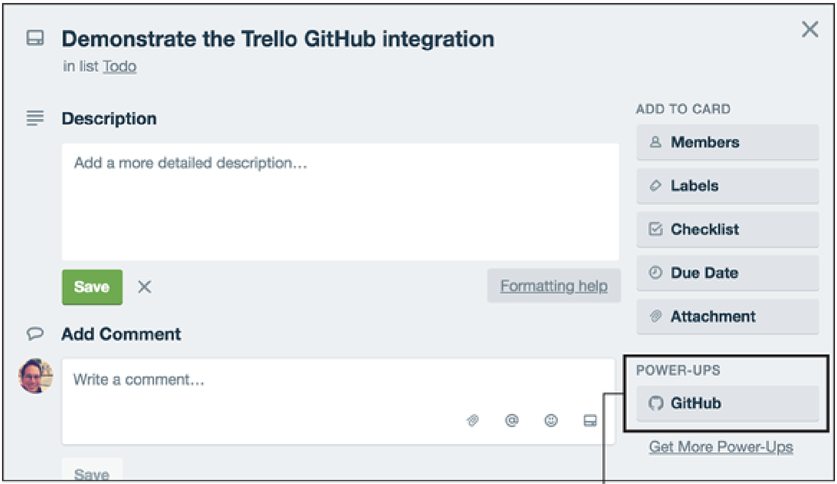
   - Attach Branch
   - Attach Commit
   - Attach Issue
   - Attach Pull Request

7. **Click Attach Issue to bring up a repository search dialog box.**

8. **Find the repository that contains the issue you want to attach to the card.**

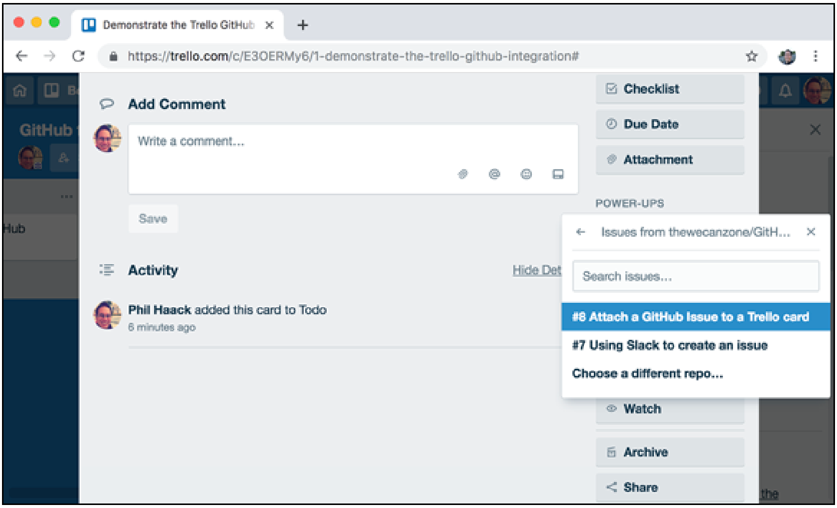
   After you select the repository, you see a list of issues, as shown in Figure 12-12. You can also search for issues.

   Clicking the issue from the drop-down list attaches it to the Trello card. After you attach the issue, the issue is displayed on the back of the Trello card.


Demonstrate the Trello GitHub integration
in list Todo
Description
Add a more detailed description…
Save
Formatting help
Add Comment
Write a comment…
Save
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachment
POWER-UPS
GitHub
Get More Power-Ups

GitHub power-up

FIGURE 12-11: Trello card with a GitHub power-up.


Demonstrate the Trello GitHub
https://trello.com/c/E3OERMy6/1-demonstrate-the-trello-github-integration#
GitHub
Hub
Add Comment
Write a comment…
Save
Activity
Hide Det
Checklist
Due Date
Attachment
POWER-UPS
Issues from thewecanzone/GitH…
Search issues…
#8 Attach a GitHub Issue to a Trello card
#7 Using Slack to create an issue
Choose a different repo…
Watch
Archive
Share
the

Phil Haack added this card to Todo
6 minutes ago

FIGURE 12-12: Selecting the issue to attach.

A Trello card may be attached to multiple GitHub items. For example, repeat the previous steps, but choose Attach Pull Request instead of Attach Issue to attach a pull request to an issue. When you are done, you see both an issue and a pull request attached to the Trello card, as shown in Figure 12-13.

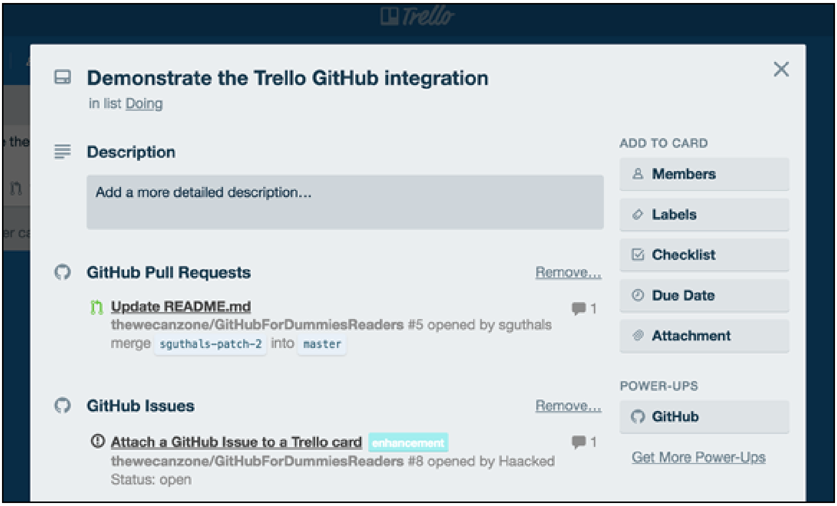

Trello
Demonstrate the Trello GitHub integration
in list Doing
Description
Add a more detailed description…
GitHub Pull Requests
Remove…
Update README.md
thewecanzone/GitHubForDummiesReaders #5 opened by sguthals
merge sguthals-patch-2 into master
1
GitHub Issues
Remove…
Attach a GitHub Issue to a Trello card
enhancement
thewecanzone/GitHubForDummiesReaders #8 opened by Haacked
Status: open
1
ADD TO CARD
Members
Labels
Checklist
Due Date
Attachment
POWER-UPS
GitHub
Get More Power-Ups

FIGURE 12-13: Trello card with an issue and pull request attached.

The front of the card shows a couple icons that indicate that this card is attached to GitHub. It shows an Octocat icon with a count of GitHub items attached to the card. It also shows pull request icon with a count to indicate the number of pull requests attached, as shown in Figure 12-14.

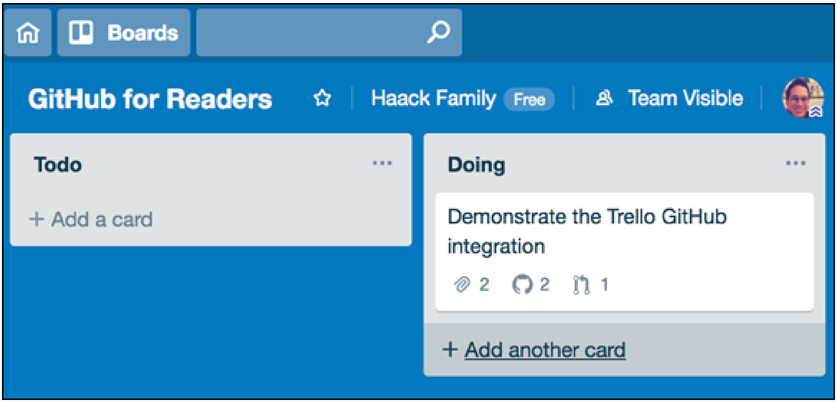
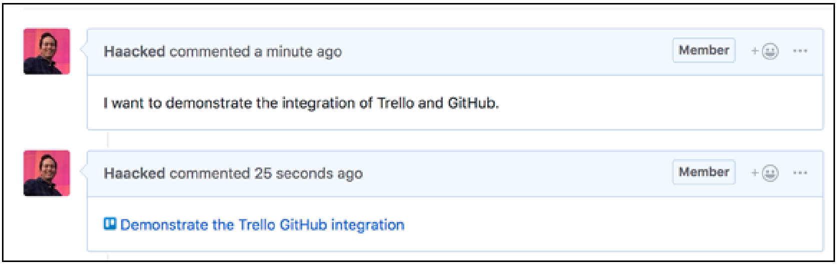

Boards
GitHub for Readers
Haack Family
Free
Team Visible
Todo
+ Add a card
Doing
Demonstrate the Trello GitHub integration
2
2
1
+ Add another card

FIGURE 12-14: Front of a card with an issue and pull request attached.

When you visit the issue or pull request on GitHub.com, you can see that the attachment is bidirectional. The GitHub issue now has a link to the trello card, as shown in Figure 12-15.



Haacked commented a minute ago

I want to demonstrate the integration of Trello and GitHub.

Haacked commented 25 seconds ago

Demonstrate the Trello GitHub integration

FIGURE 12-15: GitHub issue with a link to the Trello board.

Managing Notifications with Octobox

Earlier in this chapter, we cover a couple of integrations that bring GitHub information into other collaboration tools. GitHub integrations help teams work together.

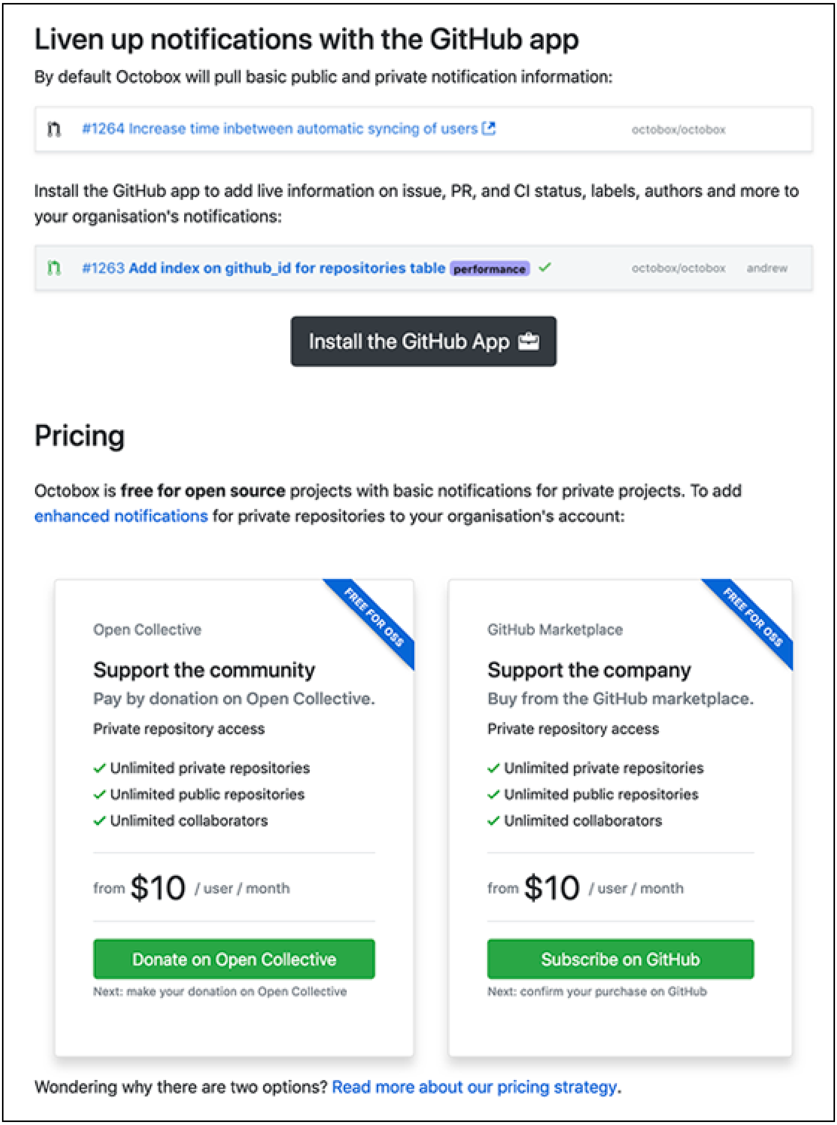
In this section, we cover a GitHub app that's a little different. It's a tool to help individuals manage the flow of GitHub notifications. As you participate in more and more GitHub repositories, the number of notifications can start to get overwhelming. Octobox provides an email client style view of your notifications.

Installing Octobox is pretty straightforward:

1. **Go to `https://octobox.io` and scroll down to the button labeled Install the GitHub App.**

   Some pricing options appear, as shown in Figure 12-16. Octobox is free for open source projects.

2. **Click Install it for free to continue with the installation process.**
3. **Authorize the application the same way you authorized Slack and Trello, earlier in this chapter.**



Liven up notifications with the GitHub app

By default Octobox will pull basic public and private notification information:

#1264 Increase time inbetween automatic syncing of users — octobox/octobox

Install the GitHub app to add live information on issue, PR, and CI status, labels, authors and more to your organisation's notifications:

#1263 Add index on github_id for repositories table performance ✓ — octobox/octobox — andrew

Pricing

Octobox is **free for open source** projects with basic notifications for private projects. To add enhanced notifications for private repositories to your organisation's account:

Open Collective

Support the community

Pay by donation on Open Collective.

Private repository access

- ✓ Unlimited private repositories
- ✓ Unlimited public repositories
- ✓ Unlimited collaborators

from $10 / user / month

Next: make your donation on Open Collective

GitHub Marketplace

Support the company

Buy from the GitHub marketplace.

Private repository access

- ✓ Unlimited private repositories
- ✓ Unlimited public repositories
- ✓ Unlimited collaborators

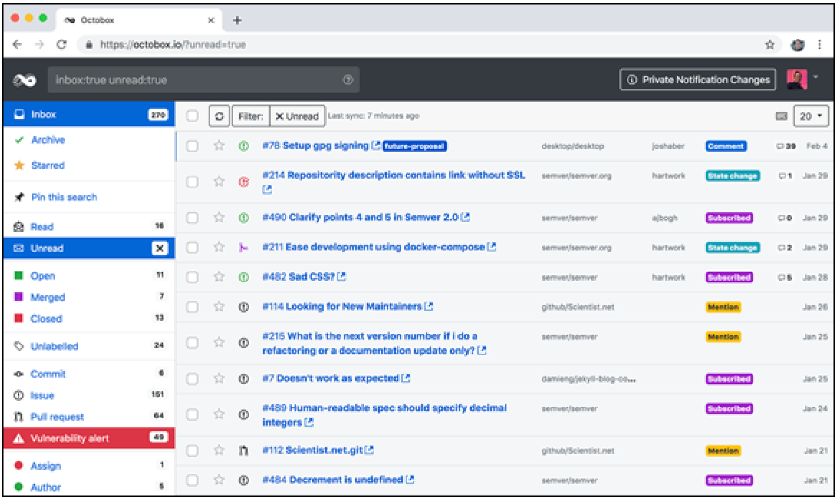
from $10 / user / month

Next: confirm your purchase on GitHub

Wondering why there are two options? Read more about our pricing strategy.

FIGURE 12-16: GitHub app download button and Pricing options for Octobox.

After the installation and authorization steps are complete, you're taken to your Octobox inbox. The first time it runs, it takes a moment to synchronize your notifications. When it's done, you should see something like Figure 12-17.


Octobox
https://octobox.io/?unread=true
inbox:true unread:true
Private Notification Changes
Inbox 270
Archive
Starred
Pin this search
Read 16
Unread
Open 11
Merged 7
Closed 13
Unlabelled 24
Commit 6
Issue 151
Pull request 64
Vulnerability alert 49
Assign 1
Author 5
Filter:
Unread
Last sync: 7 minutes ago
20

#78 Setup gpg signing future-proposal	desktop/desktop	joshaber	Comment	39	Feb 4
#214 Repository description contains link without SSL	semver/semver.org	hartwork	State change	1	Jan 29
#490 Clarify points 4 and 5 in Semver 2.0	semver/semver	ajbogh	Subscribed	0	Jan 29
#211 Ease development using docker-compose	semver/semver.org	hartwork	State change	2	Jan 29
#482 Sad CSS?	semver/semver	hartwork	Subscribed	5	Jan 28
#114 Looking for New Maintainers	github/Scientist.net		Mention		Jan 26
#215 What is the next version number if i do a refactoring or a documentation update only?	semver/semver		Mention		Jan 25
#7 Doesn't work as expected	damieng/jekyll-blog-co...		Subscribed		Jan 25
#489 Human-readable spec should specify decimal integers	semver/semver		Subscribed		Jan 24
#112 Scientist.net.git	github/Scientist.net		Mention		Jan 21
#484 Decrement is undefined	semver/semver		Subscribed		Jan 21

FIGURE 12-17: Octobox inbox for the author.

After Octobox is installed and synchronized, you can use it to manage your notifications. It allows you to search and filter your notifications by repository, organization, type, action, status, and so on. You can set Octobox to automatically synchronize on an interval in its Settings page. As the status for issues and pull requests change on GitHub, synchronizing Octobox displays those changes in Octobox. Octobox also provides archiving and muting for notifications, which is a nice way of staying on top of notifications, especially if you work on multiple active projects on GitHub.

In this chapter, we note that people collaborate on a project in many places other than GitHub. For example, people may use Slack to chat about a project and Trello to manage tasks for a project. The GitHub apps for these two products pull information from GitHub into these spaces. This is useful for those who are not on GitHub as the apps may provide important context. It also helps reduce context switching as folks may not need to constantly switch back and forth from their app and GitHub.

The last app, Octobox, serves a different purpose. It is a tool that fills in a gap in the GitHub product and makes GitHub notifications more manageable. These are just three examples of the many apps that make it possible to collaborate on GitHub in contexts outside of GitHub.

Chapter 13

GitHub Workflow Integrations

IN THIS CHAPTER

- **Exploring GitHub Integrations**
- **Using GitHub Integrations**
- **Keeping informed about GitHub Integrations**

In Chapter 12, we show you some ways that you can get information about GitHub repos in other applications. Using applications like Slack and Trello, integrated with GitHub, can especially help you with project management. In this chapter, we show you some ways you can integrate GitHub into your existing development workflow.

A lot of the integrations that we show you in this chapter are open source, which means you can track the development of new features, easily report bugs through GitHub, or even contribute to the project. It also means that each is rapidly changing, so some details may be outdated by the time you read it. What is important is that you know how to find updates and navigate each integration.

Using GitHub for Atom

Atom is a lightweight, open source editor, so you can track the progress and report issues directly on the GitHub repo where it's being developed. Atom has both a GitHub and Git integration that can help improve your development workflow when collaborating with others.

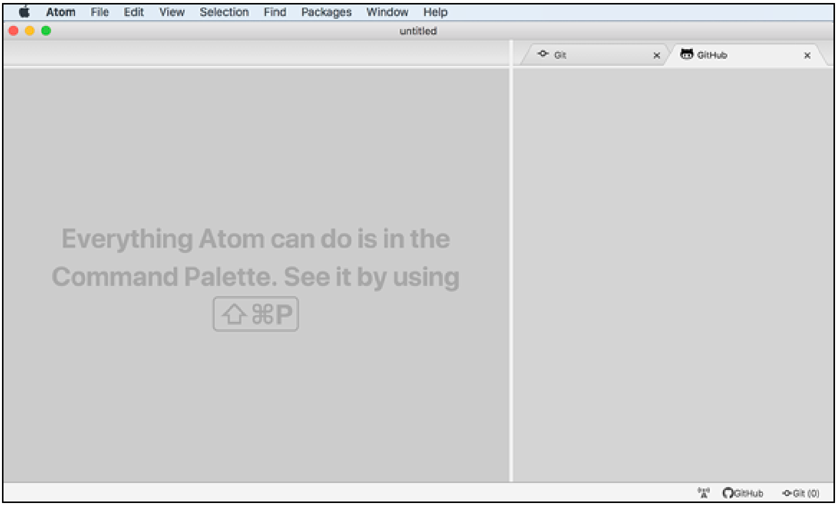
To get started, make sure you've downloaded Atom from `https://atom.io`. Then, open the GitHub tab in one of the following ways:

- Choose Packages ⇒ GitHub and then select the Toggle GitHub tab
- Click the GitHub button at the bottom right of the Atom window

- Use the keyboard shortcut Ctrl + Shift + 8.

Viewing, checking out, and creating pull requests

When you first open the GitHub tab, what you see depends on the state of the repository that you have open in Atom. Figure 13-1 shows what happens if you don't have a repository open in Atom at all. The blank screen isn't an error; there is just no project, so the GitHub tab has no actionable items or feedback for you.

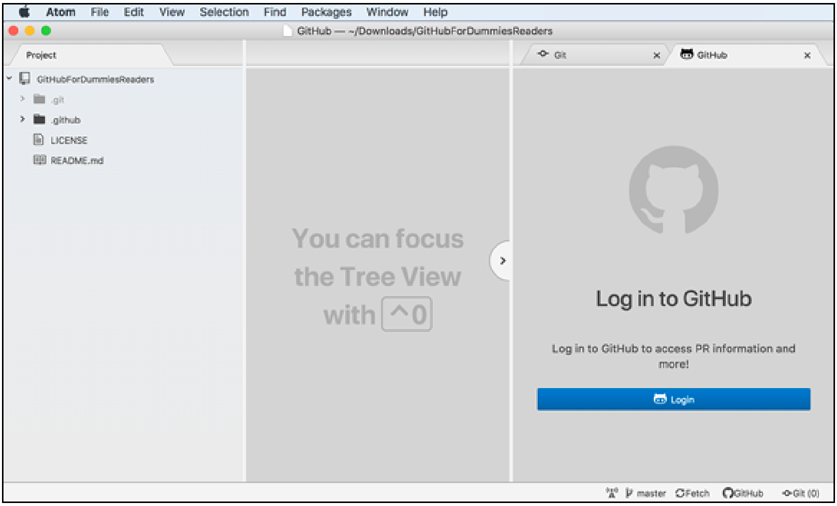


Everything Atom can do is in the Command Palette. See it by using ⇧⌘P

FIGURE 13-1: The GitHub tab when no project is open in Atom.

If you have a project open in Atom that isn't associated with a GitHub repository, then you see the same blank tab that is shown in Figure 13-1. If the project is only a local project (for example, you created the repository using `git init` or it's just a folder on your computer, but you don't have it backed up to any hosting platform like GitHub), you also have a notification at the bottom of the Atom window that indicates that no remote repository exists for this project. If the project is hosted somewhere other than GitHub, such as GitLab, you still see a blank tab like in Figure 13-1, but the bottom of the Atom window has the button for fetching, pushing, and pulling your code. This button is a feature of the Git integration and isn't specific to GitHub.

Figure 13-2 shows the login button that you see if you have a GitHub repo open in Atom. When you click the Login button, you're asked to go to `http://github.atom.io/login` to generate an authentication token to paste into the Atom window.

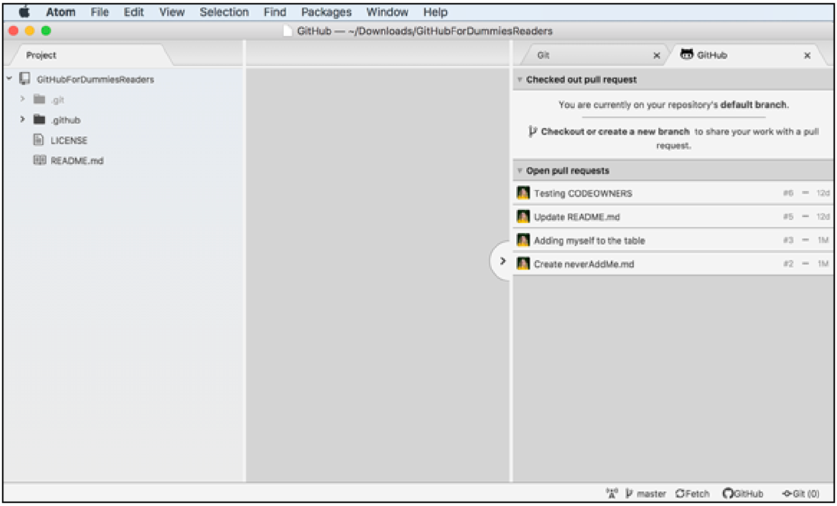

Atom File Edit View Selection Find Packages Window Help
GitHub — ~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
You can focus the Tree View with ^0
Git
GitHub
Login
master
Fetch
GitHub
Git (0)

Log in to GitHub

Log in to GitHub to access PR information and more!

FIGURE 13-2: The GitHub tab when a GitHub project is open in Atom, but you have not yet authenticated with GitHub.

After you give permission to the GitHub package for Atom to access your account, you're given a long authorization token. After you paste that token into Atom, you see a list of all the open pull requests on that repository and information about which branch you're on (see Figure 13-3).


Atom File Edit View Selection Find Packages Window Help
GitHub — ~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
Git
GitHub
master
Fetch
GitHub
Git (0)

Checked out pull request

You are currently on your repository's **default branch**.

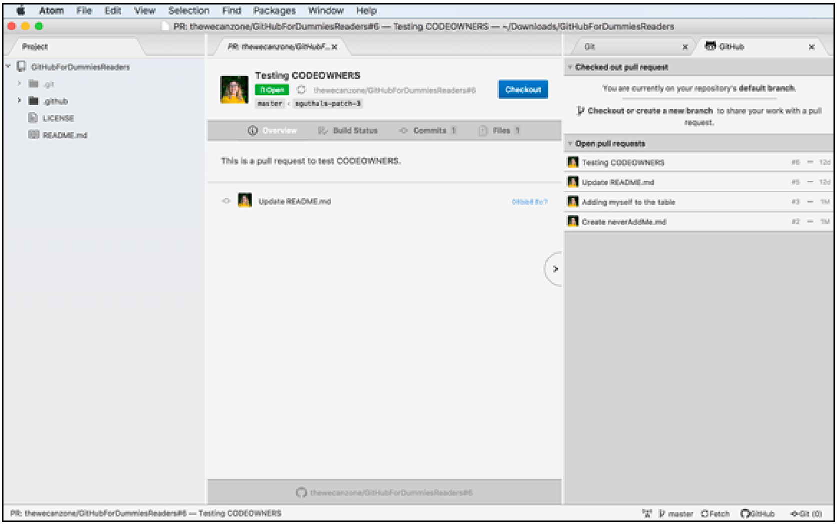
Checkout or create a new branch to share your work with a pull request.

Open pull requests

Testing CODEOWNERS	#6	12d
Update README.md	#5	12d
Adding myself to the table	#3	1M
Create neverAddMe.md	#2	1M

FIGURE 13-3: The GitHub tab showing all open pull requests on that repository.

You can check out a branch associated with a pull request by clicking the pull request and then clicking the blue Checkout button, as shown in Figure 13-4.


Atom
File
Edit
View
Selection
Find
Packages
Window
Help
PR: thewecanzone/GitHubForDummiesReaders#6 — Testing CODEOWNERS — ~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
PR: thewecanzone/GitHubF...
Testing CODEOWNERS
Open
thewecanzone/GitHubForDummiesReaders#6
master
sguthals-patch-3
Checkout
Overview
Build Status
Commits 1
Files 1
Update README.md
Git
GitHub
Checked out pull request
Checkout or create a new branch
Open pull requests
Testing CODEOWNERS
#6
12d
Update README.md
#5
12d
Adding myself to the table
#3
1M
Create neverAddMe.md
#2
1M
thewecanzone/GitHubForDummiesReaders#6
PR: thewecanzone/GitHubForDummiesReaders#6 — Testing CODEOWNERS
master
Fetch
GitHub
Git (0)

This is a pull request to test CODEOWNERS.

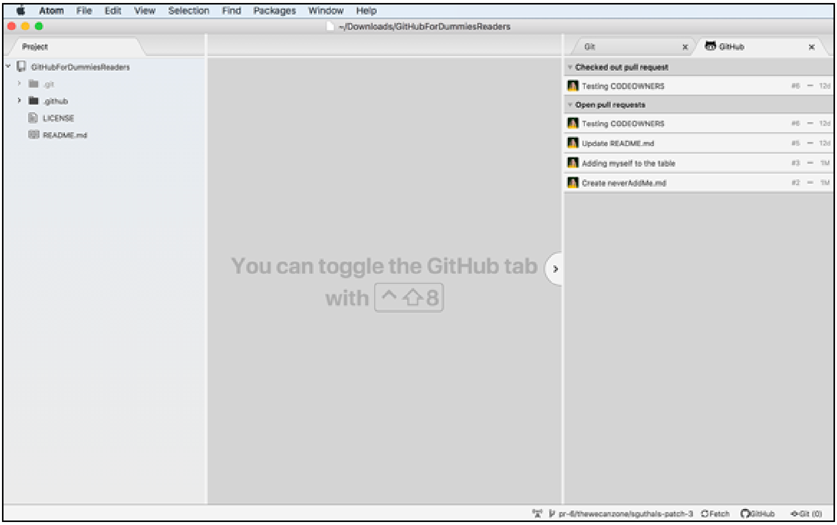
You are currently on your repository's **default branch**.

Checkout or create a new branch to share your work with a pull request.

FIGURE 13-4: The detailed view of a pull request with the Checkout button.

Notice that in Figure 13-4, multiple tabs at the top of the detailed pull request view allow you to check the build status, view the list of commits on this branch, and view the diff of the files that were changed compared to the target branch for this pull request.

When you're on a branch associated with a pull request, that pull request will be brought to the top of the GitHub tab, as shown in Figure 13-5.


Atom File Edit View Selection Find Packages Window Help
~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
Git
GitHub
Checked out pull request
Testing CODEOWNERS #6 — 12d
Open pull requests
Testing CODEOWNERS #6 — 12d
Update README.md #5 — 12d
Adding myself to the table #3 — 1M
Create neverAddMe.md #2 — 1M
pr-6/thewecanzone/sguthals-patch-3
Fetch
GitHub
Git (0)

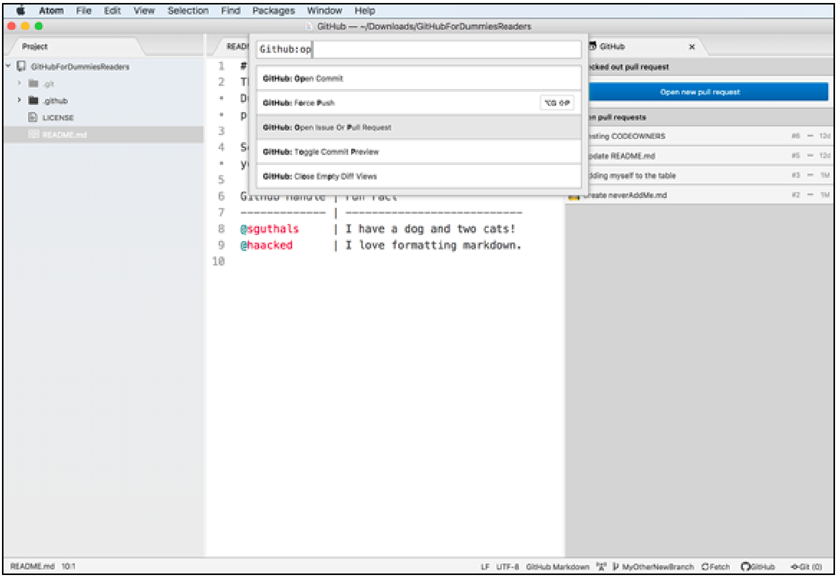
You can toggle the GitHub tab with ^⇧8

FIGURE 13-5: The GitHub tab showing the current pull request at the top.

If you create or check out a branch that isn't yet associated with a pull request, the GitHub tab adds a Publish button to the top section, above the list of open pull requests. Clicking this button publishes your branch and then takes you to the GitHub.com web page where you can create a pull request for that branch. This button replaces the Testing CODEOWNERS pull request in the Checked out pull requests section in Figure 13-5.

Viewing issues

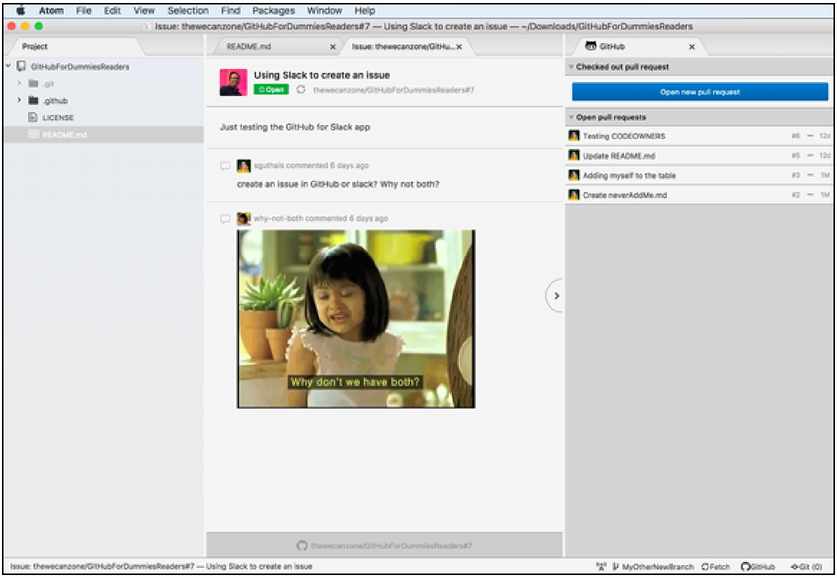
You can also view issues in Atom using a nifty Command Palette command. First, open the command palette by choosing Packages ⇒ Command Palette or by pressing ⌘ + Shift + P. From there, start typing **GitHub: Open**, and you should find the GitHub: Open Issue or Pull Request option, as shown in Figure 13-6. Choose that option.


Atom File Edit View Selection Find Packages Window Help
GitHub — ~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
Github:op
GitHub: Open Commit
GitHub: Force Push
GitHub: Open Issue Or Pull Request
GitHub: Toggle Commit Preview
GitHub: Close Empty Diff Views
GitHub
Open new pull request
Testing CODEOWNERS
Update README.md
Adding myself to the table
Create neverAddMe.md
README.md 10:1
LF UTF-8 GitHub Markdown MyOtherNewBranch Fetch GitHub Git (0)

```
@sguthals     | I have a dog and two cats!
@haacked      | I love formatting markdown.
```

FIGURE 13-6: The GitHub tab showing the current pull request at the top.

Paste in the URL to any issue (this work for pull requests also) and click the Open Issue or Pull Request button, and the issue details open in Atom with all the comments, just like on GitHub.com (see Figure 13-7).


Atom File Edit View Selection Find Packages Window Help
Issue: thewecanzone/GitHubForDummiesReaders#7 — Using Slack to create an issue — ~/Downloads/GitHubForDummiesReaders
Project
GitHubForDummiesReaders
.git
.github
LICENSE
README.md
Issue: thewecanzone/GitHu...
GitHub
Checked out pull request
Open new pull request
Open pull requests
Testing CODEOWNERS
Update README.md
Adding myself to the table
Create neverAddMe.md
Open
thewecanzone/GitHubForDummiesReaders#7
MyOtherNewBranch Fetch GitHub Git (0)

Using Slack to create an issue

Just testing the GitHub for Slack app

sguthals commented 6 days ago

create an issue in GitHub or slack? Why not both?

why-not-both commented 6 days ago

Why don't we have both?

FIGURE 13-7: GitHub issue details opened in Atom.

You can't interact with issues other than reading them, but viewing issues in Atom can still be useful if you need to reference an issue while you're working on some code. With the open issue feature, you don't have to have a browser window also open and continuously toggle between the browser and Atom.

Following the GitHub package for Atom

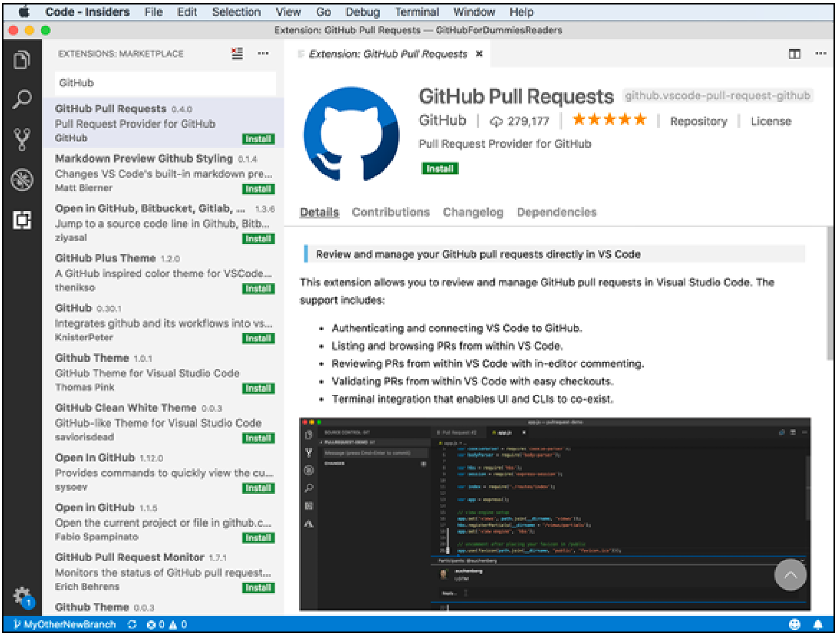
The GitHub package for Atom has extensive documentation, which you can find in the Atom Flight Manual at `https://flight-manual.atom.io/using-atom/sections/github-package`. Here you find a list of all the Git and GitHub features with screenshots and in-depth descriptions.

You can also track the progress of new or future features on the GitHub repo itself at `https://github.com/atom/github`. The GitHub package for Atom team often uses project boards, issues, pull requests, and RFCs (request for comments) to plan, design, and implement new features. For example, you can find the RFC for the pull request review features at `https://github.com/atom/github/blob/master/docs/feature-requests/003-pull-request-review.md`.

Using GitHub for Visual Studio Code

One of the fastest growing editors is Visual Studio Code (VS Code for short). With more than 2.6 million active users in the first 12 months in 2017, you have probably either used it or at least heard about it. Like Atom, VS Code is open source, which means you can see the development of new editor features at `https://github.com/microsoft/vscode`. In 2018, Microsoft and GitHub teamed up to build an open source GitHub editor extension that provides an in-editor pull request experience.

After you install VS Code, you can install this extension by going to the extension marketplace, searching for *GitHub,* and clicking the green install button for the GitHub Pull Requests extension, as shown in Figure 13-8.


Code - Insiders
File
Edit
Selection
View
Go
Debug
Terminal
Window
Help
Extension: GitHub Pull Requests — GitHubForDummiesReaders
EXTENSIONS: MARKETPLACE
Extension: GitHub Pull Requests
Details
Contributions
Changelog
Dependencies
MyOtherNewBranch
0
0

GitHub

GitHub Pull Requests 0.4.0
Pull Request Provider for GitHub
GitHub
Install

Markdown Preview Github Styling 0.1.4
Changes VS Code's built-in markdown pre...
Matt Bierner
Install

Open in GitHub, Bitbucket, Gitlab, ... 1.3.6
Jump to a source code line in Github, Bitb...
ziyasal
Install

GitHub Plus Theme 1.2.0
A GitHub inspired color theme for VSCode...
thenikso
Install

GitHub 0.30.1
Integrates github and its workflows into vs...
KnisterPeter
Install

Github Theme 1.0.1
GitHub Theme for Visual Studio Code
Thomas Pink
Install

GitHub Clean White Theme 0.0.3
GitHub-like Theme for Visual Studio Code
savlorisdead
Install

Open In GitHub 1.12.0
Provides commands to quickly view the cu...
sysoev
Install

Open in GitHub 1.1.5
Open the current project or file in github.c...
Fabio Spampinato
Install

GitHub Pull Request Monitor 1.7.1
Monitors the status of GitHub pull request...
Erich Behrens
Install

Github Theme 0.0.3

GitHub Pull Requests github.vscode-pull-request-github
GitHub | 279,177 | ★★★★★ | Repository | License
Pull Request Provider for GitHub
Install

Review and manage your GitHub pull requests directly in VS Code

This extension allows you to review and manage GitHub pull requests in Visual Studio Code. The support includes:

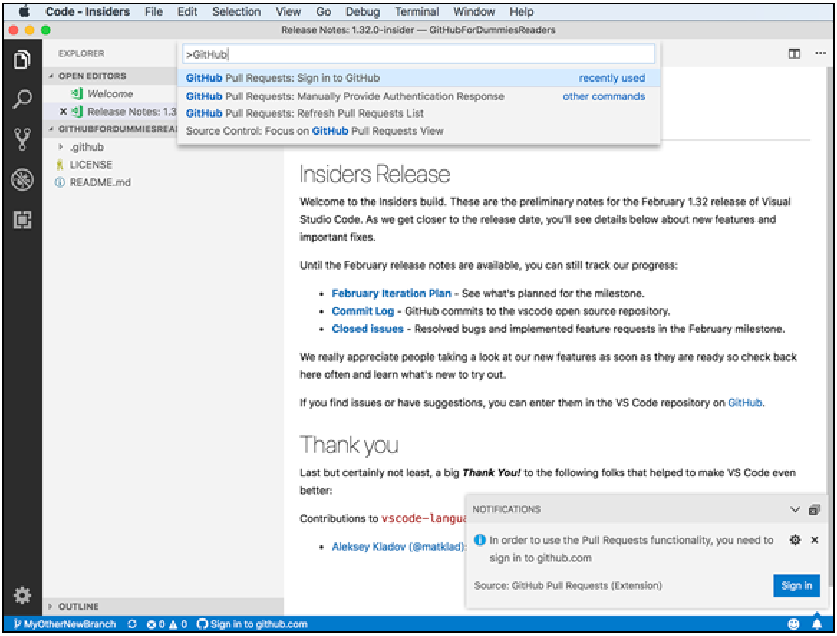
- Authenticating and connecting VS Code to GitHub.
- Listing and browsing PRs from within VS Code.
- Reviewing PRs from within VS Code with in-editor commenting.
- Validating PRs from within VS Code with easy checkouts.
- Terminal integration that enables UI and CLIs to co-exist.

FIGURE 13-8: GitHub extension for VS Code in the extension marketplace.

WARNING You have to reload the VS Code window after you install the extension. Reloading should take only a couple of seconds, and if you have any projects already open, it shouldn't be a problem.

After you install the extension, you can sign in to GitHub. You can sign in to GitHub in three ways (see Figure 13-9):

- Click the Octocat button at the bottom left of the VS Code window.
- Click the notification at the bottom right of the VS Code window.
- Use the Command Palette, which you can open by pressing ⌘ + Shift + p and searching for the sign in by typing **GitHub**.


Code - Insiders
File
Edit
Selection
View
Go
Debug
Terminal
Window
Help
Release Notes: 1.32.0-insider — GitHubForDummiesReaders
EXPLORER
OPEN EDITORS
Welcome
Release Notes: 1.3
GITHUBFORDUMMIESREA
.github
LICENSE
README.md
OUTLINE
recently used
other commands
NOTIFICATIONS
Sign in
MyOtherNewBranch
0
0
Sign in to github.com

>GitHub

GitHub Pull Requests: Sign in to GitHub

GitHub Pull Requests: Manually Provide Authentication Response

GitHub Pull Requests: Refresh Pull Requests List

Source Control: Focus on GitHub Pull Requests View

Insiders Release

Welcome to the Insiders build. These are the preliminary notes for the February 1.32 release of Visual Studio Code. As we get closer to the release date, you'll see details below about new features and important fixes.

Until the February release notes are available, you can still track our progress:

- **February Iteration Plan** - See what's planned for the milestone.
- **Commit Log** - GitHub commits to the vscode open source repository.
- **Closed issues** - Resolved bugs and implemented feature requests in the February milestone.

We really appreciate people taking a look at our new features as soon as they are ready so check back here often and learn what's new to try out.

If you find issues or have suggestions, you can enter them in the VS Code repository on GitHub.

Thank you

Last but certainly not least, a big ***Thank You!*** to the following folks that helped to make VS Code even better:

Contributions to `vscode-langu`

- Aleksey Kladov (@matklad):

In order to use the Pull Requests functionality, you need to sign in to github.com

Source: GitHub Pull Requests (Extension)

FIGURE 13-9: Three ways to start the GitHub sign-in process.

The sign-in process opens a web browser where you can authorize VS Code to access your GitHub repositories. Click through the prompts, and it redirects you back to VS Code. After you're back in VS Code, you get a pop-up notification asking whether you trust the URL; click Yes.

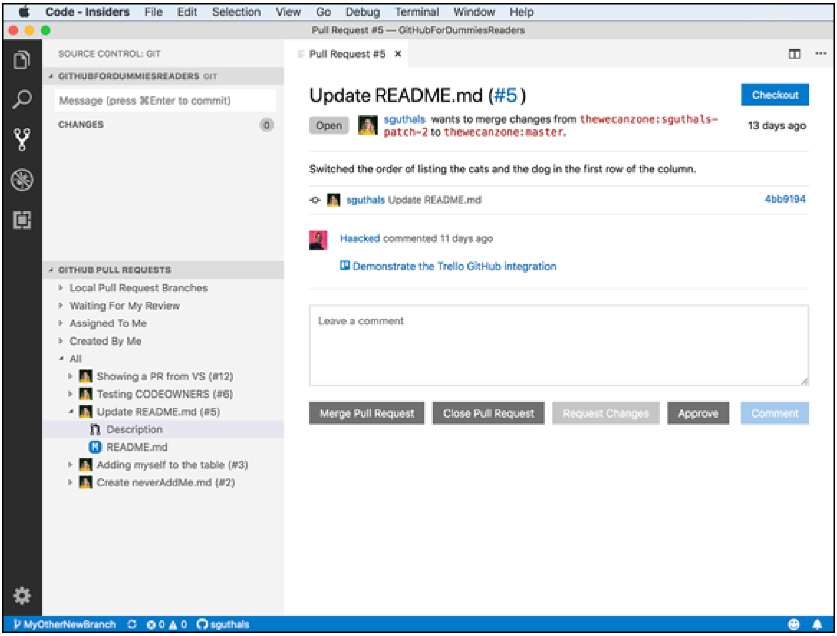
Interacting with pull requests in VS Code

After you're signed in, when you go to the Source Control tab on the left side of VS Code, you should see all the pull requests associated with this repo. Pull requests are grouped into five different sections:

- Ones that you currently have checked out
- Ones where you are marked as a reviewer
- Ones that you're assigned to
- Ones that you created
- A list of all pull requests

When you unroll a specific pull request, you see the description of the pull request, along with all the modified files.

Clicking the description of the pull request displays the description as you would see it on GitHub, as shown in Figure 13-10. From this page, you can check out the pull request, leave a comment, merge or update the pull request, or complete a review. You have the entire pull request experience right inside of VS Code!



Update README.md (#5)

sguthals wants to merge changes from thewecanzone:sguthals-patch-2 to thewecanzone:master.

Switched the order of listing the cats and the dog in the first row of the column.

sguthals Update README.md 4bb9194

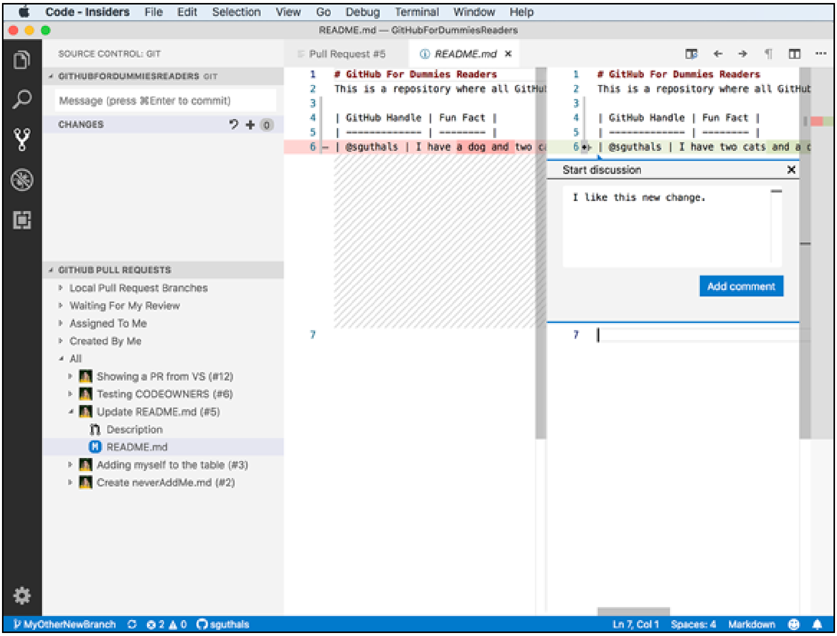
Haacked commented 11 days ago

Demonstrate the Trello GitHub integration

Code - Insiders
File
Edit
Selection
View
Go
Debug
Terminal
Window
Help
Pull Request #5 — GitHubForDummiesReaders
SOURCE CONTROL: GIT
GITHUBFORDUMMIESREADERS GIT
Message (press ⌘Enter to commit)
CHANGES
0
GITHUB PULL REQUESTS
Local Pull Request Branches
Waiting For My Review
Assigned To Me
Created By Me
All
Showing a PR from VS (#12)
Testing CODEOWNERS (#6)
Update README.md (#5)
Description
README.md
Adding myself to the table (#3)
Create neverAddMe.md (#2)
Pull Request #5
Checkout
Open
13 days ago
Leave a comment
Merge Pull Request
Close Pull Request
Request Changes
Approve
Comment
MyOtherNewBranch
0
0
sguthals

FIGURE 13-10: Interacting with pull requests in VS Code.

Another feature of this extension is the ability to add inline comments to the diff. Clicking a specific modified file shows you the side-by-side diff, just as it would look on GitHub.com. From here, you can add a comment to any of the modified lines, as shown in Figure 13-11. All these actions are reflected on GitHub.com.


Code - Insiders
File
Edit
Selection
View
Go
Debug
Terminal
Window
Help
README.md — GitHubForDummiesReaders
SOURCE CONTROL: GIT
GITHUBFORDUMMIESREADERS GIT
Message (press ⌘Enter to commit)
CHANGES
0
GITHUB PULL REQUESTS
Local Pull Request Branches
Waiting For My Review
Assigned To Me
Created By Me
All
Showing a PR from VS (#12)
Testing CODEOWNERS (#6)
Update README.md (#5)
Description
README.md
Adding myself to the table (#3)
Create neverAddMe.md (#2)
Pull Request #5
README.md
Start discussion
Add comment
MyOtherNewBranch
2
0
sguthals
Ln 7, Col 1
Spaces: 4
Markdown

```
# GitHub For Dummies Readers
This is a repository where all GitHul

| GitHub Handle | Fun Fact |
| ------------- | -------- |
| @sguthals | I have a dog and two ca
```

```
# GitHub For Dummies Readers
This is a repository where all GitHut

| GitHub Handle | Fun Fact |
| ------------- | -------- |
| @sguthals | I have two cats and a d
```

I like this new change.

FIGURE 13-11: Adding an inline comment in VS Code.

Following the GitHub for VS Code pull requests extension

Because this extension is also open source, you can follow the development, report issues, or even contribute to it on GitHub.com. Go to `https://github.com/Microsoft/vscode-pull-request-github` to find the latest features and documentation for how to use this extension to improve your development workflow.

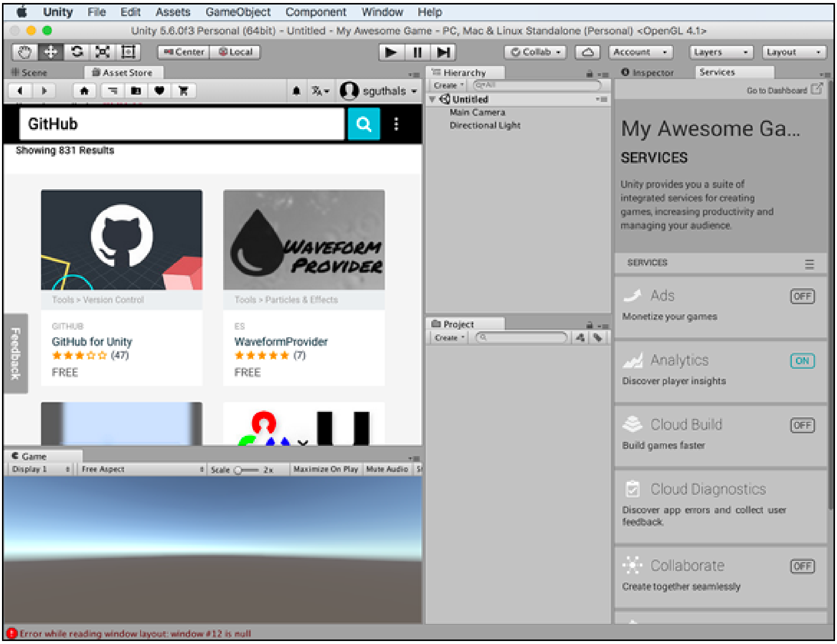
The VS Code and GitHub teams working on this extension often use a tracking issue to track what they're working on. You can find the most recent iteration plan for this project by going to the Issues tab on the GitHub repo and filtering the issues by the label iteration-plan.

Using GitHub for Unity

Unity is a cross-platform game engine that game developers use to build scenes and attach code scripts to games. You can write the code in any editor you want, or you can use MonoDevelop, the editor that is shipped with Unity.

TIP Integrated development environments (IDE), such as Visual Studio, have integrations that make writing code for your Unity game a better experience. With the full-featured experience of an IDE coupled with a clean connection back to Unity, you can be an even more effective game developer! After you create a new project in Unity, you can add the GitHub extension to your project. Unity is different from other editors. In Unity, every extension is represented as an Asset and is a part of your game, not a part of the development environment. So, when you add the GitHub extension to your Unity project, the files for that extension are a part of your game. Don't worry; they won't appear in your finished game, but they will help while you're collaboratively building it!

First, open the Asset Store in Unity by choosing Window ⇒ Asset Store. Search GitHub, and you will find the GitHub for Unity asset, as shown in Figure 13-12. It is free.

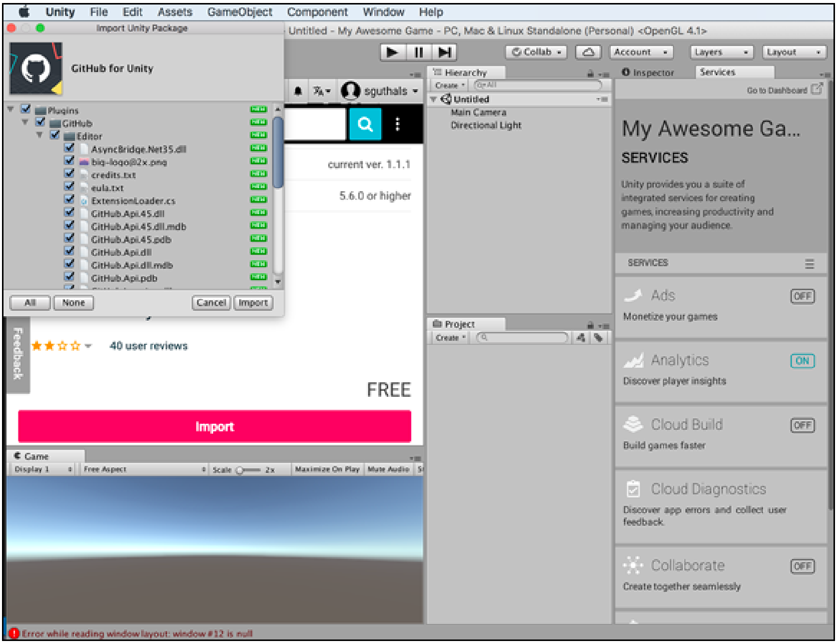

Unity File Edit Assets GameObject Component Window Help
Unity 5.6.0f3 Personal (64bit) - Untitled - My Awesome Game - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Center
Local
Collab
Account
Layers
Layout
Scene
Asset Store
sguthals
GitHub
Showing 831 Results
Tools > Version Control
GITHUB
GitHub for Unity
(47)
FREE
WAVEFORM PROVIDER
Tools > Particles & Effects
ES
WaveformProvider
(7)
FREE
Feedback
Game
Display 1
Free Aspect
Scale 2x
Maximize On Play
Mute Audio
Hierarchy
Create
Untitled
Main Camera
Directional Light
Project
Create
Inspector
Services
Go to Dashboard
My Awesome Ga...
SERVICES
SERVICES
Ads
OFF
Monetize your games
Analytics
ON
Discover player insights
Cloud Build
OFF
Build games faster
Cloud Diagnostics
Discover app errors and collect user feedback.
Collaborate
OFF
Create together seamlessly

Unity provides you a suite of integrated services for creating games, increasing productivity and managing your audience.

Error while reading window layout: window #12 is null

FIGURE 13-12: GitHub for Unity in the Asset Store.

Click the GitHub for Unity asset, scroll down, and click the big pink Import button. A pop-up window with all the files that will be added to your project appears, as shown in Figure 13-13. Click Import.


Unity
File
Edit
Assets
GameObject
Component
Window
Help
Import Unity Package
GitHub for Unity
Plugins
GitHub
Editor
AsyncBridge.Net35.dll
big-logo@2x.png
credits.txt
eula.txt
ExtensionLoader.cs
GitHub.Api.45.dll
GitHub.Api.45.dll.mdb
GitHub.Api.45.pdb
GitHub.Api.dll
GitHub.Api.dll.mdb
GitHub.Api.pdb
NEW
All
None
Cancel
Import
Untitled - My Awesome Game - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Collab
Account
Layers
Layout
sguthals
current ver. 1.1.1
5.6.0 or higher
Feedback
40 user reviews
FREE
Import
Game
Display 1
Free Aspect
Scale
2x
Maximize On Play
Mute Audio
Hierarchy
Create
Untitled
Main Camera
Directional Light
Project
Create
Inspector
Services
Go to Dashboard
My Awesome Ga...
SERVICES
SERVICES
Ads
OFF
Monetize your games
Analytics
ON
Discover player insights
Cloud Build
OFF
Build games faster
Cloud Diagnostics
Discover app errors and collect user feedback.
Collaborate
OFF
Create together seamlessly

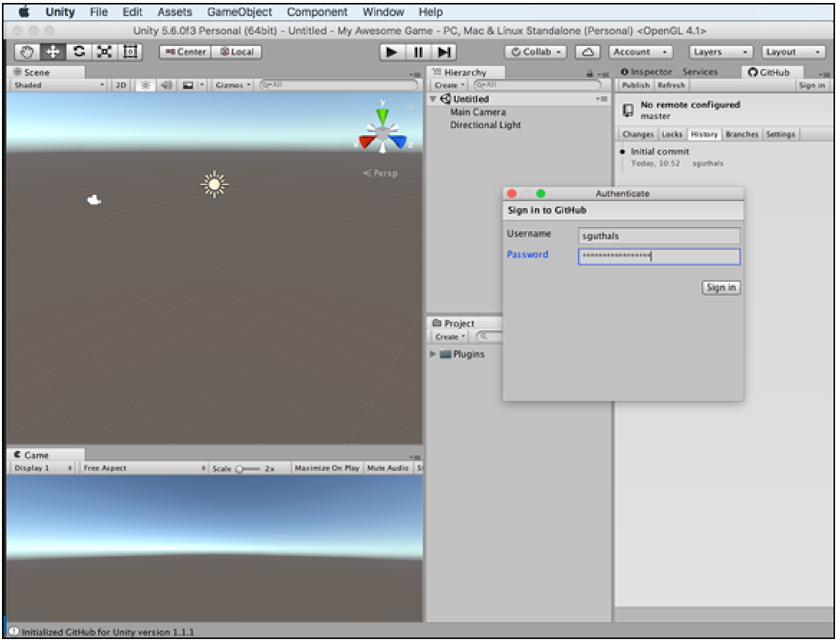
Unity provides you a suite of integrated services for creating games, increasing productivity and managing your audience.

Error while reading window layout: window #12 is null

FIGURE 13-13: Installing the GitHub for Unity extension.

Using GitHub for Unity in Unity

After you install the extension, you see a new menu item in the Window menu called GitHub. If you choose GitHub, a new tab opens in Unity (see in the background of Figure 13-14). Click the Initialize a git repository for this project button in the center of the new tab, and the GitHub tab updates with additional tabs related to git actions. At the top right of the GitHub tab, you can click the Sign in button to sign in to your GitHub account, shown in Figure 13-14. If you have two-factor authentication setup, you are asked for your 2fa code.


Unity File Edit Assets GameObject Component Window Help
Center Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
Persp
Hierarchy
Create
Untitled
Main Camera
Directional Light
Inspector
Services
GitHub
Publish
Refresh
Sign in
Changes
Locks
History
Branches
Settings
Authenticate
Sign in to GitHub
Sign in
Project
Plugins
Game
Display 1
Free Aspect
Scale 2x
Maximize On Play
Mute Audio

Unity 5.6.0f3 Personal (64bit) - Untitled - My Awesome Game - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>

No remote configured
master

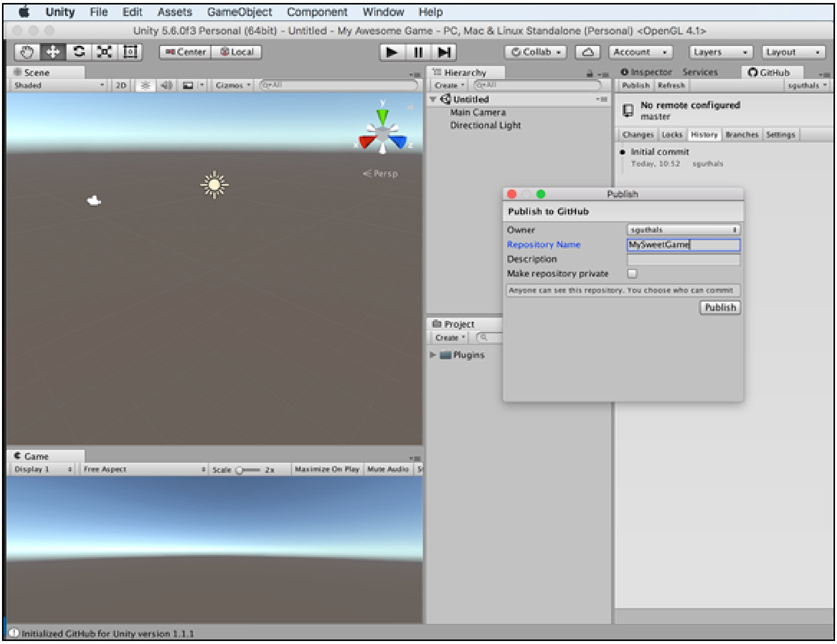
Initial commit
Today, 10:52 sguthals

Username: sguthals
Password: ****************

Initialized GitHub for Unity version 1.1.1

FIGURE 13-14: Signing in to GitHub inside of Unity.

When you're ready, you can publish your project to GitHub by clicking the Publish button at the top left of the GitHub tab. In the Publish dialog box, shown in Figure 13-15, choose which GitHub organization should own this project, put in a repository name, add a description, and choose to make it public or private.


Unity File Edit Assets GameObject Component Window Help
Unity 5.6.0f3 Personal (64bit) - Untitled - My Awesome Game - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Center
Local
Collab
Account
Layers
Layout
Scene
Shaded
2D
Gizmos
Persp
Hierarchy
Create
Untitled
Main Camera
Directional Light
Inspector
Services
GitHub
Publish
Refresh
sguthals
No remote configured
master
Changes
Locks
History
Branches
Settings
Initial commit
Today, 10:52 sguthals
Project
Plugins
Game
Display 1
Free Aspect
Scale 2x
Maximize On Play
Mute Audio
Initialized GitHub for Unity version 1.1.1

Publish

Publish to GitHub

Owner: sguthals

Repository Name: MySweetGame

Description:

Make repository private ☐

Anyone can see this repository. You choose who can commit

Publish

FIGURE 13-15: Publishing your project to GitHub.com.

After the project is published, you can continue to work on your project just like you would any other coding project. Items that you may find useful include

- **Changes** where all the recent changes are listed. You can add changes to a commit and then commit them to the branch that you're on.
- **Locks** where you can lock certain files and assets so that only you can modify them. This lock becomes extremely important if you're making changes to a game scene and you don't want someone else to also make changes to that scene and overwrite your work.
- **History** is a history of all the commits for this project.
- **Branches** where you can see all the branches on this repo, create a new branch, and switch branches.
- **Settings** where you can change the remote path of your repository, change the path to git that this extension is using, and even update your git username or email. It's recommended that you don't modify these settings unless you know what you are doing. The extension will set these up properly for you.

- **Fetch/Pull/Push/Refresh** is at the top of the GitHub tab and contains four buttons that sync your GitHub repo with your local project.
- **GitHub Username** is at the top right of the GitHub tab where you see your username and go to your GitHub.com profile or sign out of GitHub inside of Unity.

Following the GitHub for Unity extension

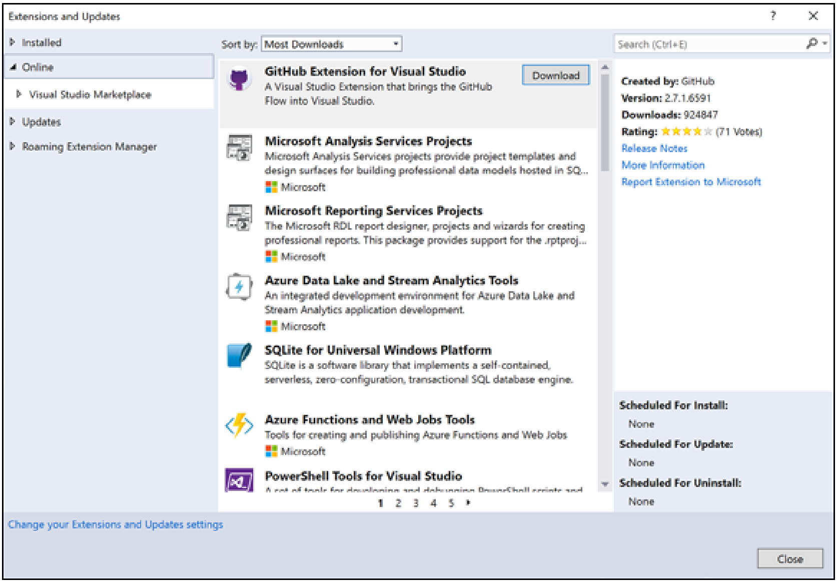
The GitHub for Unity extension is open source, so you can track development, report issues, or even contribute to it via the GitHub repo at `https://github.com/github-for-unity/Unity`. It's also important to always keep the extension up to date. A pop-up window appears in Unity whenever a new version is released, but you can also check on the Unity Asset store or directly download the latest version all from `https://unity.github.com`.

The team working on this extension is fairly small, so if you're interested in contributing to the project, we'd highly encourage you to give it a go! You can always check out the issues on the repo on GitHub.com for ones labeled Help Wanted or Good First Issue.

Using GitHub for Visual Studio

Visual Studio is different from Visual Studio Code. Visual Studio is an IDE and is a full-featured application to support developers in writing code. Both VS Code and Atom are editors that have an extensive list of extensions that a developer can add to the editor to create the experience they need. Visual Studio has a version for Mac and a version for Windows. This section talks only about the version for Windows because it's the original and more full-featured version and the one that has the GitHub integration.

After you install Visual Studio, choose Tools ⇒ Extensions and Updates. A pop-up window appears with all the extensions you currently have installed, plus the marketplace of additional extensions. If you click Online, the top choice is the GitHub Extension for Visual Studio extension, as shown in Figure 13-16.


Extensions and Updates
?
Installed
Online
Visual Studio Marketplace
Updates
Roaming Extension Manager
Sort by:
Search (Ctrl+E)
Download
1 2 3 4 5
Change your Extensions and Updates settings
Close

Sort by: Most Downloads

GitHub Extension for Visual Studio
A Visual Studio Extension that brings the GitHub Flow into Visual Studio.

Microsoft Analysis Services Projects
Microsoft Analysis Services projects provide project templates and design surfaces for building professional data models hosted in SQ...
Microsoft

Microsoft Reporting Services Projects
The Microsoft RDL report designer, projects and wizards for creating professional reports. This package provides support for the .rptproj...
Microsoft

Azure Data Lake and Stream Analytics Tools
An integrated development environment for Azure Data Lake and Stream Analytics application development.
Microsoft

SQLite for Universal Windows Platform
SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.

Azure Functions and Web Jobs Tools
Tools for creating and publishing Azure Functions and Web Jobs
Microsoft

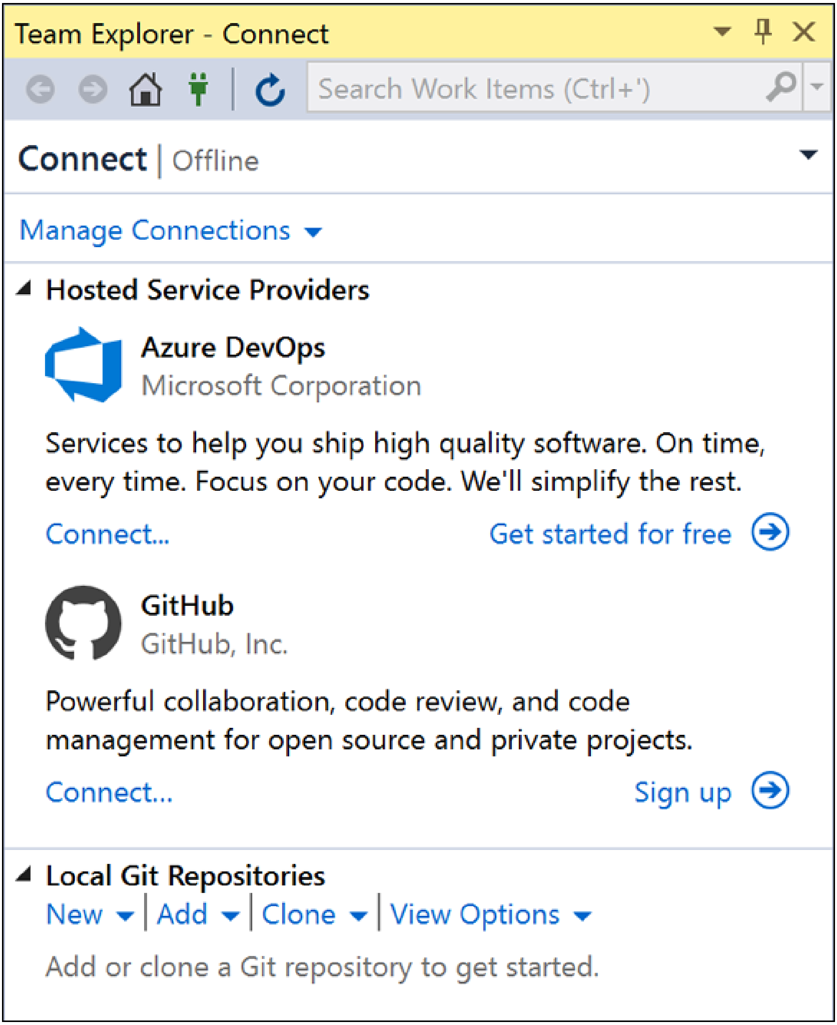
PowerShell Tools for Visual Studio

Created by: GitHub
Version: 2.7.1.6591
Downloads: 924847
Rating: ★★★★☆ (71 Votes)
Release Notes
More Information
Report Extension to Microsoft

Scheduled For Install:
None
Scheduled For Update:
None
Scheduled For Uninstall:
None

FIGURE 13-16: The GitHub for Visual Studio extension in the Visual Studio Marketplace.

Click the Download button and close Visual Studio. When Visual Studio closes, the VSIX Installer starts and ask whether it can modify Visual Studio. When you click yes, then modify, the extension begins to install. After it installs, click the Connect link in the Team Explorer tab and connect to your GitHub account, as shown in Figure 13-17. When you click the Connect link, a pop-up window asks you to sign in to GitHub. If you have two-factor authentication set up, you’re also asked for your 2fa code.


Team Explorer - Connect
Search Work Items (Ctrl+')
Connect | Offline
Manage Connections
Hosted Service Providers
Azure DevOps
Microsoft Corporation

Services to help you ship high quality software. On time, every time. Focus on your code. We'll simplify the rest.

Connect...
Get started for free
GitHub
GitHub, Inc.

Powerful collaboration, code review, and code management for open source and private projects.

Connect...
Sign up
Local Git Repositories
New
Add
Clone
View Options

Add or clone a Git repository to get started.

FIGURE 13-17: The Team Explorer pane with the GitHub Connect section.

Once connected, you can clone a repo, create a new repo, or sign out from GitHub, all from the Connect page on the Team Explorer pane.

Viewing, creating, and reviewing pull requests in Visual Studio

When you have the project open in Visual Studio that is connected to a GitHub repo, you see additional project options on the home page of the Team Explorer pane, as shown in Figure 13-18.

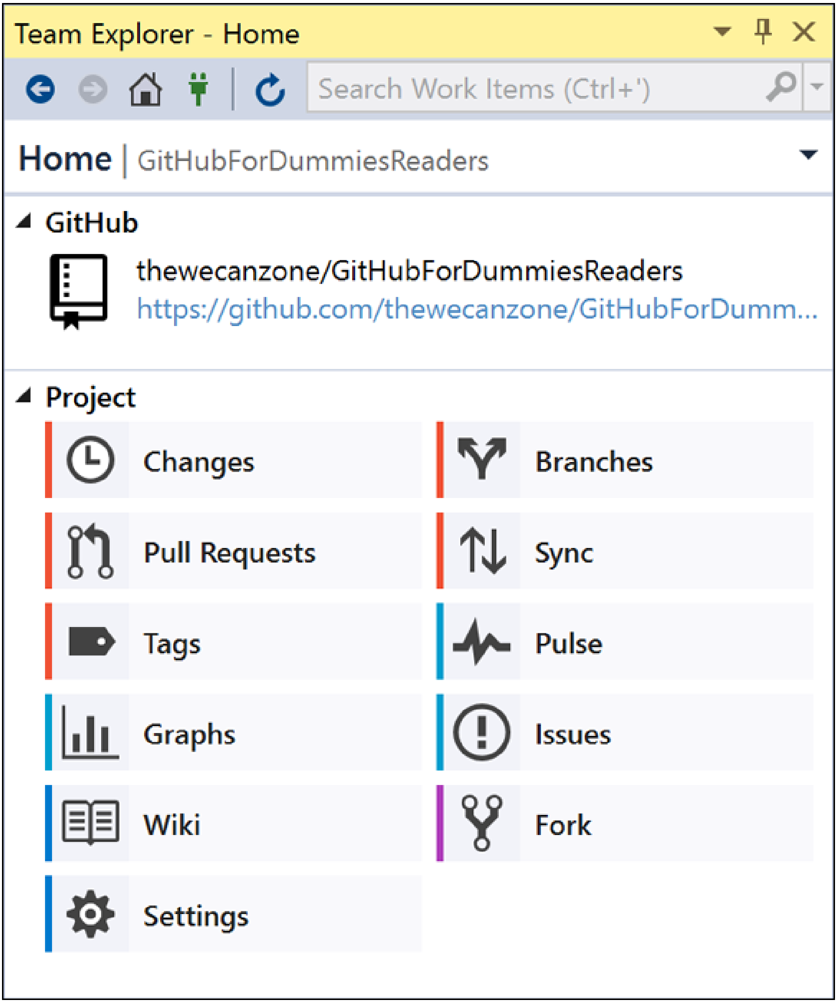
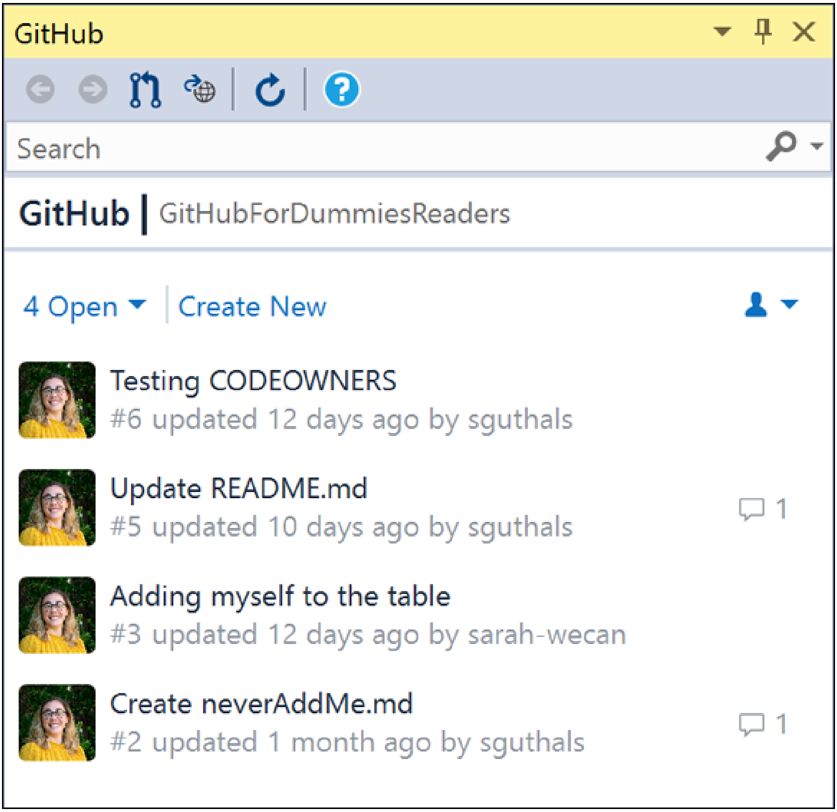

Team Explorer - Home
Search Work Items (Ctrl+')
Home | GitHubForDummiesReaders
GitHub
thewecanzone/GitHubForDummiesReaders
https://github.com/thewecanzone/GitHubForDumm...
Project
Changes
Branches
Pull Requests
Sync
Tags
Pulse
Graphs
Issues
Wiki
Fork
Settings

FIGURE 13-18: The Team Explorer pane with additional GitHub project options.

Clicking Pull Requests opens the GitHub pane where all the pull requests on this repo are listed, as shown in Figure 13-19. At the top of the list, you can choose to see all the open pull requests, closed pull requests, or just all pull requests. You can also sort by author.



Testing CODEOWNERS
#6 updated 12 days ago by sguthals

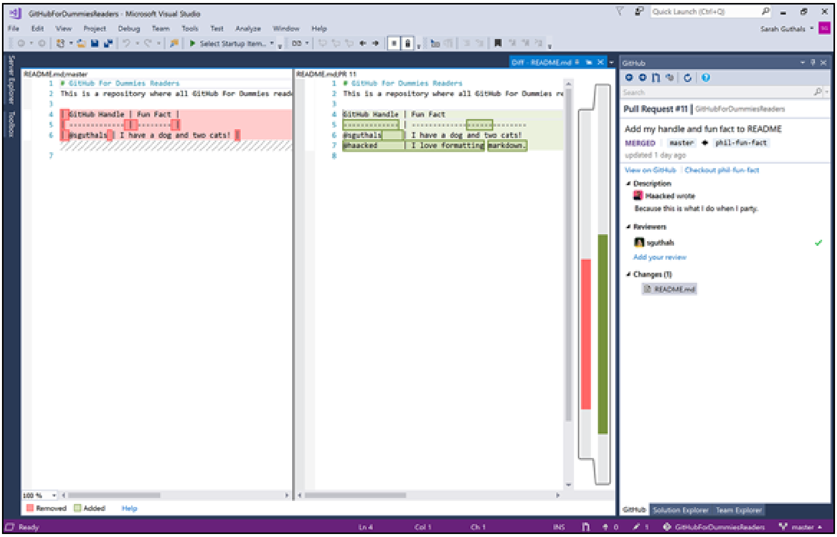
Update README.md
#5 updated 10 days ago by sguthals
1

Adding myself to the table
#3 updated 12 days ago by sarah-wecan

Create neverAddMe.md
#2 updated 1 month ago by sguthals
1

FIGURE 13-19: A list of pull requests on the GitHub pane.

If you double-click one of the pull requests, the details open. From here, you can see the description, the target and base branch, the current state, the list of reviews, and the list of files changed. You can also click the View on GitHub link to open a browser window to this pull request on github.com. Click the Checkout <*branch-name*> button to check out the branch associated with this pull request. Click the Add your review link to add your own review with inline and overall comments. When you add a review, you can mark it as comment only, approve it, or request changes. If you double-click one of the changed files, the diff opens in the editor area. If you hover over one of the changed lines, you can add an inline comment, very similar to how it is done in VS Code (refer to Figure 13-11). You can see all of this in Figure 13-20.


GitHubForDummiesReaders - Microsoft Visual Studio
File
Edit
View
Project
Debug
Team
Tools
Test
Analyze
Window
Help
Select Startup Item...
Quick Launch (Ctrl+Q)
Sarah Guthals
Server Explorer
Toolbox
Diff - README.md
README.md;master
README.md;PR 11
GitHub
Search
100 %
Removed
Added
Help
GitHub
Solution Explorer
Team Explorer
Ready
Ln 4
Col 1
Ch 1
INS
GitHubForDummiesReaders
master

README.md;master

```
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies read

| GitHub Handle | Fun Fact |
| ------------- | -------- |
| @sguthals | I have a dog and two cats! |
```

README.md;PR 11

```
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies re

GitHub Handle | Fun Fact
------------- | ----------------
@sguthals     | I have a dog and two cats!
@haacked      | I love formatting markdown.
```

Pull Request #11 | GitHubForDummiesReaders

Add my handle and fun fact to README

MERGED | master ← phil-fun-fact

updated 1 day ago

View on GitHub | Checkout phil-fun-fact

Description

Haacked wrote

Because this is what I do when I party.

Reviewers

sguthals ✓

Add your review

Changes (1)

README.md

FIGURE 13-20: Interacting with a pull request in Visual Studio.

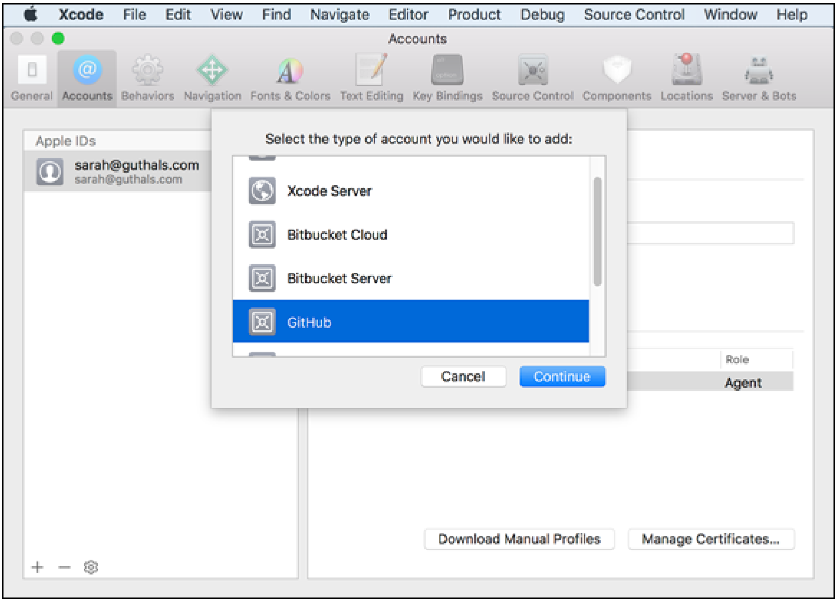
Following the GitHub for Visual Studio extension

The GitHub for Visual Studio extension is open source. You can follow the development, report issues, or even contribute to it on GitHub.com. Go to `https://github.com/github/visualstudio` to find the latest features and documentation for how to use the GitHub for Visual Studio extension to improve your development workflow.

The GitHub team working on the extension often use a tracking issue to track what it's working on. You can find the most recent tracking issue for this project by going to the Issues tab on the GitHub repo and filtering the issues by the Next Release label.

Using GitHub for XCode

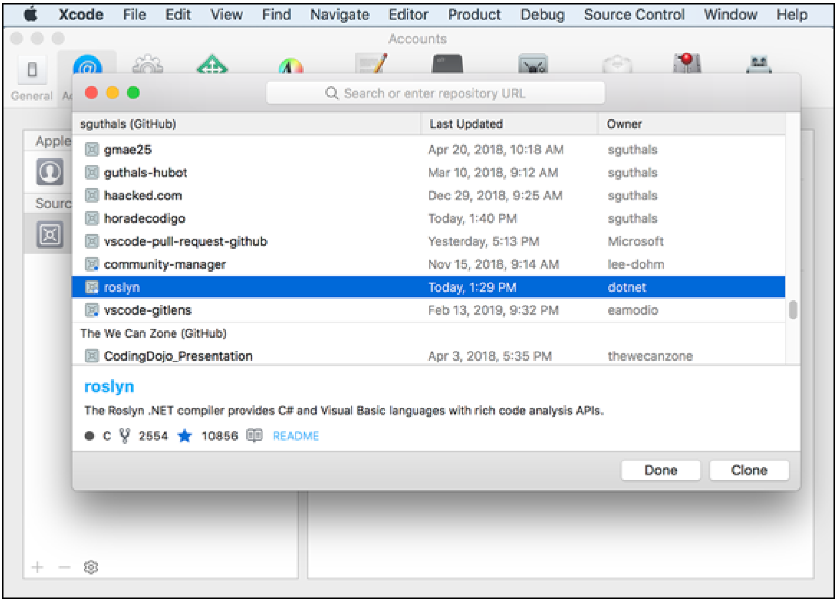
Apple has developed an integration for GitHub in XCode. After you install XCode, you can sign in with GitHub by choosing XCode ⇒ Preferences to open the Accounts tab. Click the plus sign in the bottom left corner, choose GitHub, and click Continue, as shown in Figure 13-21. Follow the prompts to sign in to your GitHub account. If you have two-factor authentication set up, you are asked for your 2fa code.


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
Help
Accounts
General
Accounts
Behaviors
Navigation
Fonts & Colors
Text Editing
Key Bindings
Source Control
Components
Locations
Server & Bots
Apple IDs
sarah@guthals.com
sarah@guthals.com
Xcode Server
Bitbucket Cloud
Bitbucket Server
GitHub
Cancel
Continue
Role
Agent
Download Manual Profiles
Manage Certificates...
+
—

Select the type of account you would like to add:

FIGURE 13-21: Signing in to GitHub inside of XCode.

Now when you choose Source Control ⇒ Clone, your GitHub repos load below the box where you can insert a URL to clone. When you click one, you get information, such as the primary language used, the number of forks and starts for this repo, and a quick link to the README, as shown in Figure 13-22.


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
Help
Accounts
General
Apple
Sourc
Search or enter repository URL
Done
Clone

sguthals (GitHub)	Last Updated	Owner
gmae25	Apr 20, 2018, 10:18 AM	sguthals
guthals-hubot	Mar 10, 2018, 9:12 AM	sguthals
haacked.com	Dec 29, 2018, 9:25 AM	sguthals
horadecodigo	Today, 1:40 PM	sguthals
vscode-pull-request-github	Yesterday, 5:13 PM	Microsoft
community-manager	Nov 15, 2018, 9:14 AM	lee-dohm
roslyn	Today, 1:29 PM	dotnet
vscode-gitlens	Feb 13, 2019, 9:32 PM	eamodio
The We Can Zone (GitHub)		
CodingDojo_Presentation	Apr 3, 2018, 5:35 PM	thewecanzone

roslyn

The Roslyn .NET compiler provides C# and Visual Basic languages with rich code analysis APIs.

● C 2554 ★ 10856 README

FIGURE 13-22: List of GitHub repositories you have access to clone from inside XCode.

If you have a file open in XCode, the Source Control menu displays additional actions you can perform, such as Commit, Push, Pull, and Fetch and Refresh Status.

This extension isn't open source, so it's best to keep up with the latest Apple developer news to know what new features may be available. You can read about all the source control integrations that Apple releases at `https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/UsingSourceCodeControl.html`.

Using GitHub for IntelliJ

The IntelliJ IDE from JetBrains has GitHub pull request support for any GitHub repo you have open. After you install IntelliJ, you can choose to clone a git project from the Start menu, as shown in Figure 13-23.

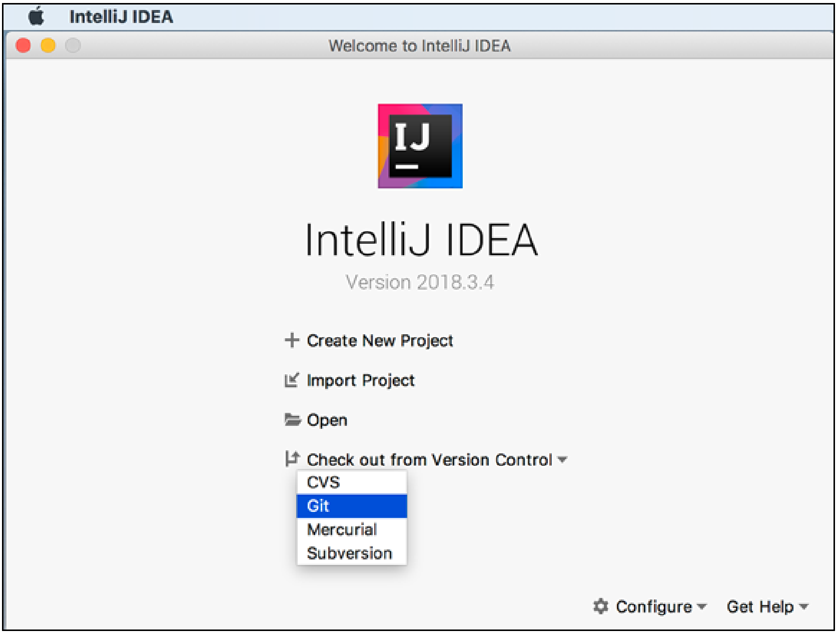
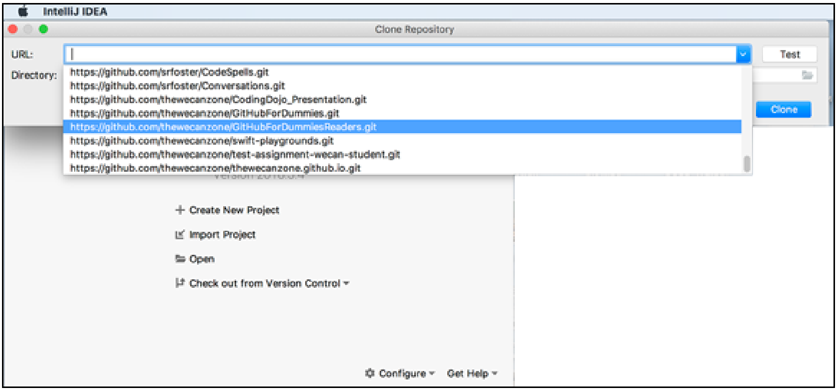

IntelliJ IDEA
Welcome to IntelliJ IDEA
IJ
IntelliJ IDEA
Version 2018.3.4
+ Create New Project
Import Project
Open
Check out from Version Control
CVS
Git
Mercurial
Subversion
Configure
Get Help

FIGURE 13-23: Clone from a git project to get started integrating GitHub into IntelliJ.

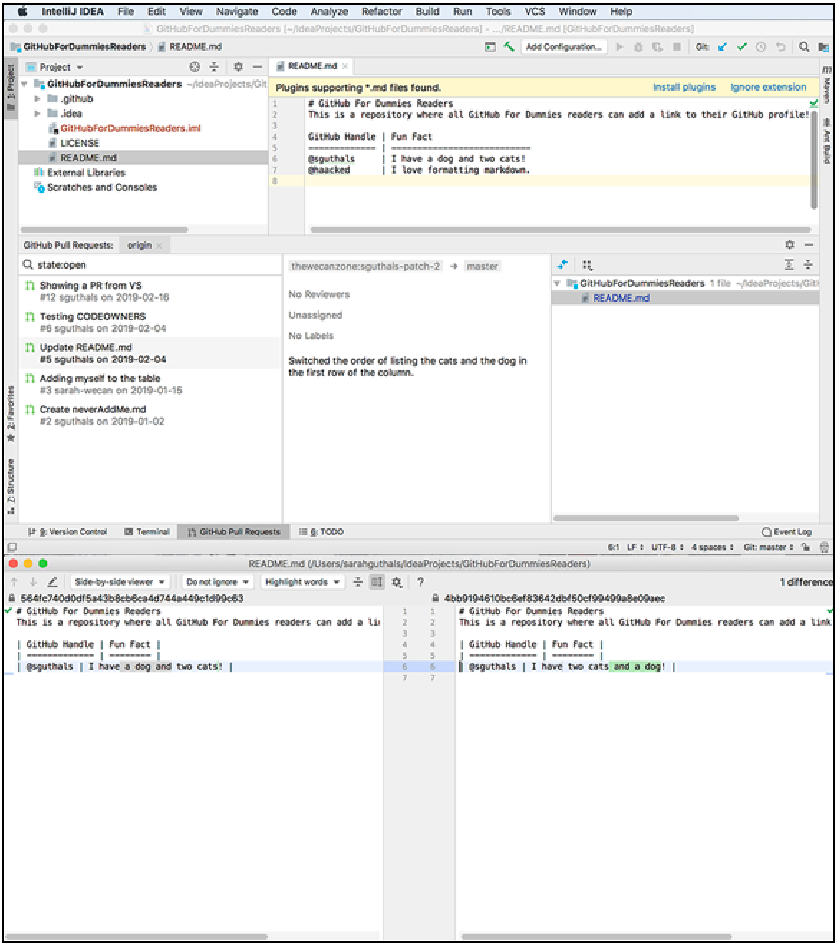
A pop-up window asks for a URL. At the bottom of this window, click the Sign in to GitHub button. You're asked for your GitHub credentials. If you have two-factor authentication set up, you are also asked for your 2fa code. After you have successfully logged in, all the GitHub repositories that you have access to appear in the drop-down list, as shown in Figure 13-24.


IntelliJ IDEA
Clone Repository
URL:
Directory:
Test
Clone
+ Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help

https://github.com/srfoster/CodeSpells.git
https://github.com/srfoster/Conversations.git
https://github.com/thewecanzone/CodingDojo_Presentation.git
https://github.com/thewecanzone/GitHubForDummies.git
https://github.com/thewecanzone/GitHubForDummiesReaders.git
https://github.com/thewecanzone/swift-playgrounds.git
https://github.com/thewecanzone/test-assignment-wecan-student.git
https://github.com/thewecanzone/thewecanzone.github.io.git

FIGURE 13-24: List of GitHub repositories you have access to clone from inside IntelliJ.

After you choose a repository and click Clone, a series of pop-up windows prompt you to create the IntelliJ project. When the project is open in IntelliJ, you can open the GitHub pull request preview by choosing VCS ⇒ Git ⇒ View Pull Requests. A new section opens in the IntelliJ window with a list of the open pull requests. If you click one, the description and list of changed files opens. If you double-click one of the changed files, a diff of that file opens in a new window (see Figure 13-25).



```
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

GitHub Handle | Fun Fact
============= | ========================
@sguthals     | I have a dog and two cats!
@haacked      | I love formatting markdown.
```

GitHub Pull Requests: origin

state:open

- Showing a PR from VS
  #12 sguthals on 2019-02-16
- Testing CODEOWNERS
  #6 sguthals on 2019-02-04
- Update README.md
  #5 sguthals on 2019-02-04
- Adding myself to the table
  #3 sarah-wecan on 2019-01-15
- Create neverAddMe.md
  #2 sguthals on 2019-01-02

thewecanzone:sguthals-patch-2 → master

No Reviewers

Unassigned

No Labels

Switched the order of listing the cats and the dog in the first row of the column.

README.md (/Users/sarahguthals/IdeaProjects/GitHubForDummiesReaders)

1 difference

```
564fc740d0df5a43b8cb6ca4d744a449c1d99c63
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies readers can add a li

| GitHub Handle | Fun Fact |
| ============= | ======== |
| @sguthals | I have a dog and two cats! |
```

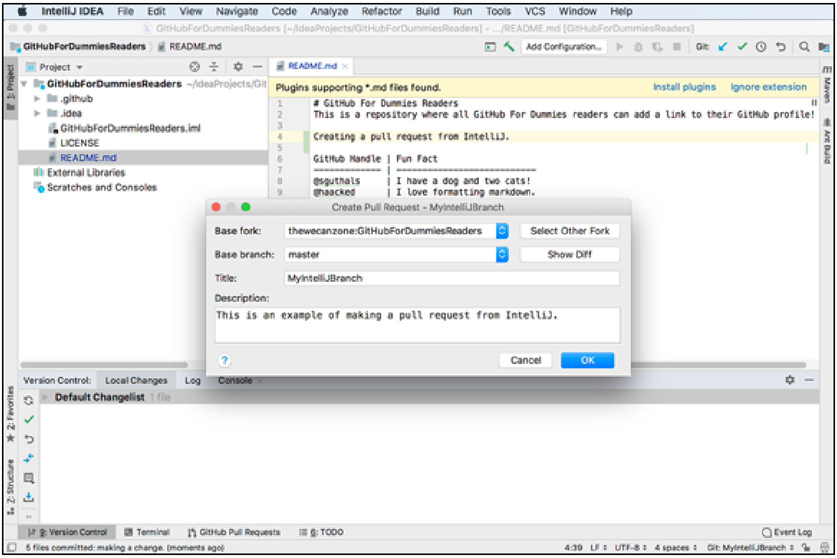
```
4bb9194610bc6ef83642dbf50cf99499a8e09aec
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies readers can add a link

| GitHub Handle | Fun Fact |
| ============= | ======== |
| @sguthals | I have two cats and a dog! |
```

IntelliJ IDEA File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
GitHubForDummiesReaders
README.md
Project
.github
.idea
GitHubForDummiesReaders.iml
LICENSE
External Libraries
Scratches and Consoles
Plugins supporting *.md files found.
Install plugins
Ignore extension
Add Configuration...
Version Control
Terminal
GitHub Pull Requests
TODO
Event Log
Side-by-side viewer
Do not ignore
Highlight words

FIGURE 13-25: Viewing a pull request from inside of IntelliJ.

Lastly, you can create a pull request from inside of IntelliJ as well. If you're on a new branch and have already made some changes and committed them to your branch, you can choose VSC ⇒ Git ⇒ Create

Pull Request. In the pop-up window, shown in Figure 13-26, specify the base and target branch, title, and description of your pull request.


IntelliJ IDEA File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
GitHubForDummiesReaders [~/IdeaProjects/GitHubForDummiesReaders] - .../README.md [GitHubForDummiesReaders]
GitHubForDummiesReaders
README.md
Add Configuration...
Git:
Project
GitHubForDummiesReaders ~/IdeaProjects/Git
.github
.idea
GitHubForDummiesReaders.iml
LICENSE
README.md
External Libraries
Scratches and Consoles
Plugins supporting *.md files found.
Install plugins
Ignore extension
Version Control:
Local Changes
Log
Console
Default Changelist 1 file
9: Version Control
Terminal
GitHub Pull Requests
6: TODO
Event Log
4:39 LF UTF-8 4 spaces Git: MyIntelliJBranch
Select Other Fork
Show Diff
Cancel
OK

```
# GitHub For Dummies Readers
This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

Creating a pull request from IntelliJ.

GitHub Handle | Fun Fact
------------- | ------------------------------
@sguthals     | I have a dog and two cats!
@haacked      | I love formatting markdown.
```

Create Pull Request - MyIntelliJBranch

Base fork: thewecanzone:GitHubForDummiesReaders

Base branch: master

Title: MyIntelliJBranch

Description:

This is an example of making a pull request from IntelliJ.

5 files committed: making a change. (moments ago)

FIGURE 13-26: Creating a pull request from inside of IntelliJ.

This GitHub pull request feature is embedded in the IntelliJ IDE, so it’s best to follow the IntelliJ blog and documentation for up-to-date information on the development of this feature.

Chapter 14

Personalizing GitHub

IN THIS CHAPTER

- **Using Chrome Extensions to extend GitHub**
- **Setting up Probot to personalize GitHub**
- **Using GitHub Actions to customize GitHub**

When GitHub created the Atom text editor, it set out to build the most hackable text editor ever. The tagline for Atom is "A hackable text editor for the 21st Century." Atom focused on that tagline because it recognized that software developers can be very particular about their tools. Developers have a lot of opinions about how they work and invest a lot of time personalizing their tools to work just the way they want.

The same is true for GitHub.com itself. GitHub offers an extensive API that lets developers write tools to interact with GitHub data in a myriad of ways. Also, since GitHub.com runs in a browser, anyone can build browser extensions to customize the experience of using GitHub.

In this chapter, we look at some of the available ways to personalize your use of GitHub.

Using Browser Extensions

Browser extensions can completely customize the experience of using a browser. You can find extensions for every possible scenario you can think of.

In this section, we look at a few useful extensions that work with GitHub. Some of these extensions are available for multiple browsers, but we focus on Google Chrome extensions for brevity.

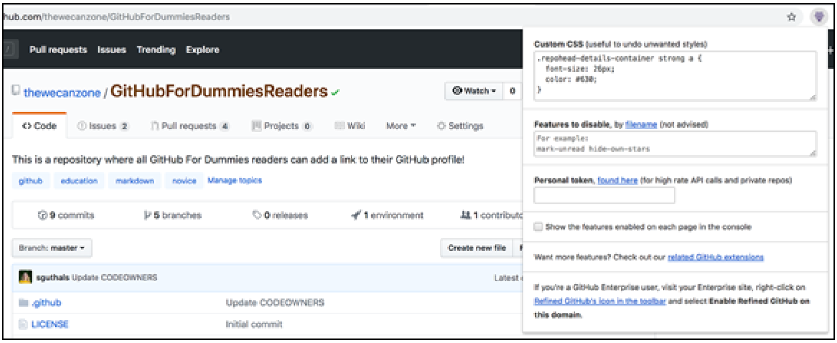
For a more comprehensive list of browser extensions that work with GitHub, check out my Awesome browser extensions for GitHub repository at `https://github.com/stefanbuck/awesome-browser-extensions-for-github`.

Refining GitHub

The Refined GitHub extension is an open source extension that simplifies the GitHub interface and adds some useful features to GitHub.com. The extension is available for the Chrome, Firefox, and Opera browsers.

You can find the source code at `https://github.com/sindresorhus/refined-github`. The README has a link to install the extension. Note that when you install the extension, you grant it the ability to read and change your data on `api.github.com`, `gist.github.com`, and `github.com`. It can also modify data you copy and paste.

After you install the extension, you should see an Octocat icon to the right of the address bar. This icon provides some light customization options. One option lets you define some custom CSS specific to GitHub.com. In Figure 14-1, you can see I added some custom CSS to make the repository name larger and dark red. Note that after you change the CSS, you have to refresh the page to see your changes in effect.


hub.com/thewecanzone/GitHubForDummiesReaders
Pull requests
Issues
Trending
Explore
thewecanzone / GitHubForDummiesReaders
Watch ▾ 0
Code
Issues 2
Pull requests 4
Projects 0
Wiki
More ▾
Settings
github
education
markdown
novice
Manage topics
9 commits
5 branches
0 releases
1 environment
1 contributo
Branch: master ▾
Create new file
sguthals Update CODEOWNERS
Latest
.github
Update CODEOWNERS
LICENSE
Initial commit

This is a repository where all GitHub For Dummies readers can add a link to their GitHub profile!

Custom CSS (useful to undo unwanted styles)

```
.repohead-details-container strong a {
  font-size: 26px;
  color: #630;
}
```

Features to disable, by filename (not advised)

```
For example:
mark-unread hide-own-stars
```

Personal token, found here (for high rate API calls and private repos)

☐ Show the features enabled on each page in the console

Want more features? Check out our related GitHub extensions

If you're a GitHub Enterprise user, visit your Enterprise site, right-click on Refined GitHub's icon in the toolbar and select **Enable Refined GitHub on this domain.**

FIGURE 14-1: Configuring the Refined GitHub extension.

Changing the CSS is just a parlor trick compared to the many other enhancements Refined GitHub brings with it.

WARNING Browser extensions work by manipulating the generated HTML of a website. It often looks for known *landmarks* in the pages (such as an HTML element with a known ID) and then adds its own UI elements, removes elements, or changes elements. However, it does this all outside of the code running on the server that actually generates the HTML. What this means is that if a website such as GitHub.com changes their HTML mark-up, an extension feature could stop working temporarily until the authors update their extension to adapt to the new change. So if any of these features stop working, try again later after the extension is updated.

The following is a small sampling of enhancements that are on as a default when you have this extension installed:

- **Mark issues and pull requests as unread.** This enhancement adds a Mark as unread button to the Notifications section of an issue or pull request. Click it to mark the issue or pull request unread, which puts it back in your notifications list.
- **Stop the page jumps from recently pushed branches.** Normally, when you push a new branch, the home page of a repository displays a list of recently pushed branches in a yellow bar above the list of code files. The sudden appearance of this list can cause the rest of the page to jump down to make room for the bar. Refined GitHub displays the list of recently pushed branches in the top right overlaying the location where the Fork button may be. By displaying the branches in an overlay, Refined GitHub ensures that the page doesn't jump down.
- **Adds option to wait for successful checks.** If you have continuous integration (CI) set up for a repository, it can take a while after someone pushes a pull request before all the checks are completed. The CI might be running static analysis or a linter, unit tests, integration tests, and so on. It can be annoying to wait for all of those processes to complete after you've reviewed some code and are ready to merge the pull request. Refined GitHub adds a check box that lets you indicate that it should go ahead and merge the pull request after the checks are complete and successful.
- **Reaction avatars show who reacted to a comment.** Typically a reaction comment shows only the reaction and the count for that reaction. With Refined GitHub, you can see who all gave a specific

reaction.

Refined GitHub contains many more enhancements big and small. The list here is just the tip of the iceberg. In Figure 14-1, notice that there is also a place to disable features that are a part of the extension, as well as a place to put your GitHub personal access token so that the extension can work on private repos as well.

Taking a GitHub selfie

As an open source project maintainer, I am ecstatic when someone comes along and submits a pull request for a project. I'm happy when someone opens an issue that identifies a problem I didn't know about. I feel gratitude for the folks that take time out of their day to help out my project.

And sure, I could use words to communicate my gratitude, but they always seem to fall short of my true feelings. If the old saying that a picture is worth a thousand words is true, how many words is an animated gif worth?

GitHub Selfie is an open source browser extension (`https://github.com/thieman/github-selfies`) that adds a Selfie button to the comment field that lets you take a selfie using your computer's camera. You can choose to take a still picture, but where's the fun in that? It also provides an option to take an animated gif.

Figure 14-2 shows one of the authors using the GitHub Selfie to take a ridiculous self-portrait.

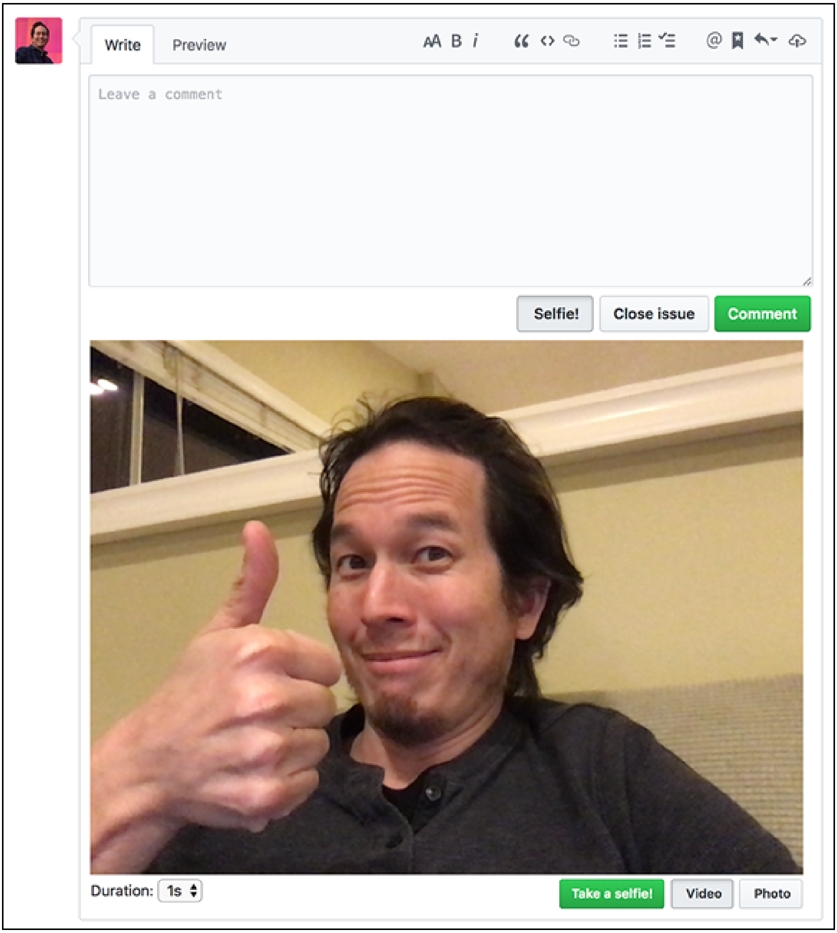

Write
Preview
AA B i
Leave a comment
Selfie!
Close issue
Comment
Duration: 1s
Take a selfie!
Video
Photo

FIGURE 14-2: Taking a short video with GitHub Selfie.

Adding a selfie to express your gratitude is a small detail, but adds a nice warm personal touch to working together with people from all over the world.

GitHub Apps and Probot

Chapter 13 covers a few integrations that connect GitHub to other applications, such as Slack and Trello. What those integrations have in common is they were implemented as GitHub apps.

Apps on GitHub let you extend GitHub in powerful ways. GitHub apps are web applications that can respond to events on GitHub. These event subscriptions are called *web hooks*. When an event occurs on GitHub that the app is interested in, GitHub makes an HTTP request to the app with information about the event. The app can then respond to that event in some manner, often resulting in a call back to GitHub via the GitHub API.

In this section, we walk through building a simple GitHub App that brings a bit of levity to your issue discussions. There's an old meme in the form of an animated gif with a little girl who asks the question, "Why don't we have both?" The typical application of this meme is in response to a question that presents a false dichotomy. In other words, when someone presents a question with two choices, someone might respond with this image.

In this section, you create a GitHub application that will automatically do this as a fun exercise.

Introducing Probot

GitHub apps are web applications that need to listen to HTTP requests. You have a lot of important details to get just right when building an HTTP request, such as what is the format of the data posted to the app? All these details can be confusing and time consuming to get correct when building a GitHub app from scratch. Knowing where to start is difficult.

GitHub's Probot framework comes in handy when getting started with a GitHub app. Probot handles much of the boilerplate and nitpicky details of building a GitHub app. It is a framework for building GitHub apps using Node.js. It provides many convenience methods for listening to GitHub events and for calling into the GitHub API.

Probot makes it easy to build a GitHub app, but it doesn't solve the problem of where to host the app.

Hosting the app

A GitHub app can take many forms. It could be a Node.js app running in Heroku, an ASP.NET Core app running in Azure, a Django app running in Google Cloud — it doesn't matter. It just needs to be persistent and available via the public Internet so that GitHub can reach it with event payloads.

Setting all that up can be time consuming, so for our purposes, we use Glitch to implement a quick and dirty GitHub app.

Introducing Glitch

Glitch (`https://glitch.com`) is a hosting platform for web applications that removes a lot of the friction with getting a web app up and running. Any app you create in Glitch is live on the web from the beginning. You don't have to think about how you plan to deploy the code because any change you make is auto saved and automatically deployed.

Glitch focuses on the community aspect of building apps. Every file can be edited by multiple people in real-time, in the same way you might edit a document in Google Docs. And every project can be remixed by clicking a button. This encourages a lot of sharing of code and learning from each other, which comes in handy when we build our own GitHub app.

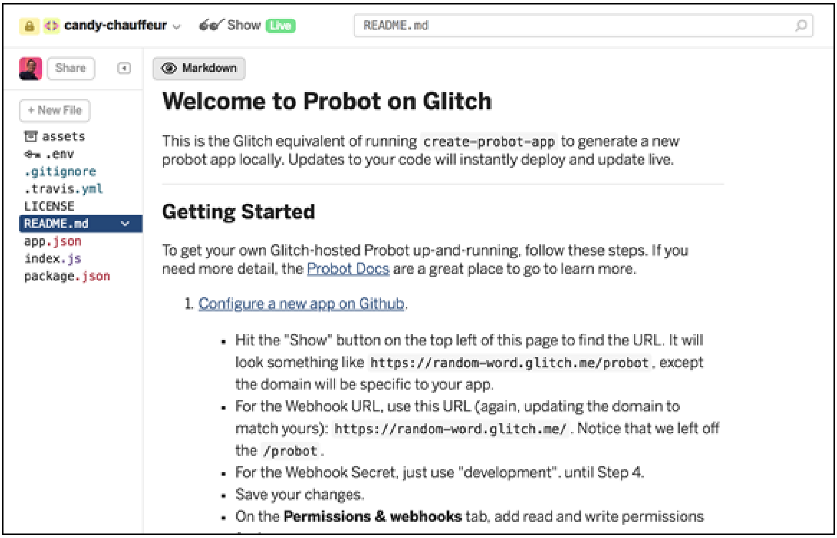
Before you continue, make sure to create an account on Glitch if you don't have one already.

Creating a Probot Glitch app

After you have a rough understanding of Probot and have a Glitch account set up, you can build a Probot app on Glitch. Glitch lets you remix existing apps, and the good news is Glitch already has a Probot app that you can remix. This means you can create your Probot app with one click and a few customizations.

To create your app, type the following URL into your browser: `https://glitch.com/edit/#!/remix/probot-hello-world`.

This command creates a brand new app in Glitch based on the `probot-hello-world` example with a randomly generated URL, as shown in Figure 14-3. As you can see, my app is called `candy-chaffeur`.



Welcome to Probot on Glitch

This is the Glitch equivalent of running `create-probot-app` to generate a new probot app locally. Updates to your code will instantly deploy and update live.

Getting Started

To get your own Glitch-hosted Probot up-and-running, follow these steps. If you need more detail, the Probot Docs are a great place to go to learn more.

1. Configure a new app on Github.
    - Hit the "Show" button on the top left of this page to find the URL. It will look something like `https://random-word.glitch.me/probot`, except the domain will be specific to your app.
    - For the Webhook URL, use this URL (again, updating the domain to match yours): `https://random-word.glitch.me/`. Notice that we left off the `/probot`.
    - For the Webhook Secret, just use "development". until Step 4.
    - Save your changes.
    - On the **Permissions & webhooks** tab, add read and write permissions

FIGURE 14-3: The README page of a Glitch Probot app.

The left pane shows the list of files in your application. The `README.md` file contains step-by-step instructions to set up the `hello-world` Probot app. Follow these instructions carefully to set up the sample GitHub app.

One of the instructions mentions running the following command:

```
cat my-app-name.2018-06-20.private-key.pem | pbcopy
```

WARNING The purpose of the previous command is to copy the contents of your private key file into the clipboard so that you can paste it into the Glitch file. However, this command works only on a Mac. On Windows, you would run the following command (changing the file name to match yours):

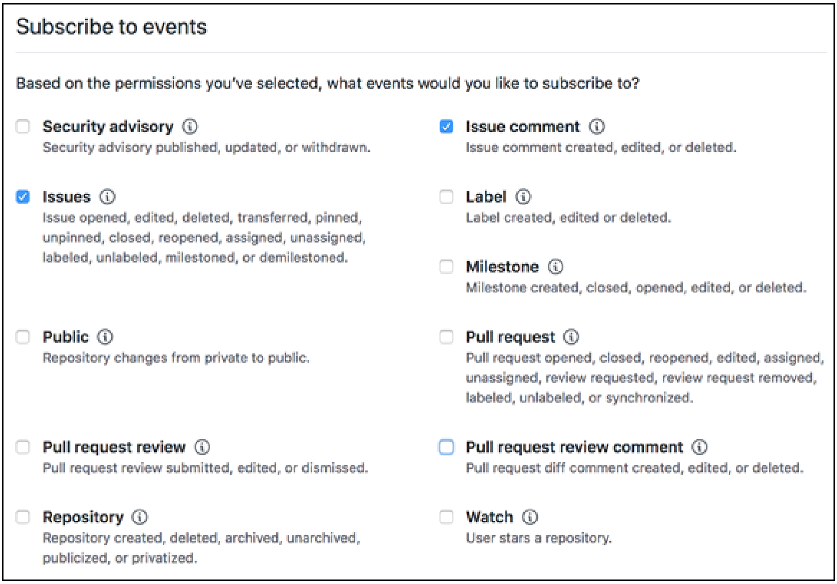
```
type my-app-name.2018-06-20.private-key.pem | clip
```

When you are done, install the app on a repository that you own and then create a new issue. A few seconds later, you should see a comment created by your bot with the words “Hello World!”.

Customizing the app

After you create a Probot app in Glitch and install it on GitHub, you can customize how the app responds to issue comments. When you followed the steps in the README, you subscribed to issue events. These events do not include when new comments are created. We need to also subscribe to issue comments.

You can see a list of your apps at `https://github.com/settings/apps`. Click the Edit button to navigate to your app. Then in the left navigation, click Permissions & events and scroll down to the Subscribe to events section. Check the Issue comment check box, as shown in Figure 14-4.



Subscribe to events

Based on the permissions you've selected, what events would you like to subscribe to?

☐ **Security advisory** ⓘ
Security advisory published, updated, or withdrawn.

☑ **Issue comment** ⓘ
Issue comment created, edited, or deleted.

☑ **Issues** ⓘ
Issue opened, edited, deleted, transferred, pinned, unpinned, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, or demilestoned.

☐ **Label** ⓘ
Label created, edited or deleted.

☐ **Milestone** ⓘ
Milestone created, closed, opened, edited, or deleted.

☐ **Public** ⓘ
Repository changes from private to public.

☐ **Pull request** ⓘ
Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, or synchronized.

☐ **Pull request review** ⓘ
Pull request review submitted, edited, or dismissed.

☐ **Pull request review comment** ⓘ
Pull request diff comment created, edited, or deleted.

☐ **Repository** ⓘ
Repository created, deleted, archived, unarchived, publicized, or privatized.

☐ **Watch** ⓘ
User stars a repository.

FIGURE 14-4: Updating the event subscriptions for the GitHub app.

Click the Save changes button at the bottom to complete these changes.

Now you need to change your Glitch app to listen to new issue comments and respond appropriately. Edit the `index.js` file and replace the contents of the file with the following code:

```
module.exports = (app) => {

  // Listens to new issue comments
  app.on('issue_comment.created', async context => {
    // Retrieves the comment text
    const message = context.payload.comment.body
    if (message.indexOf(' or ') > -1) {
      const params = context.issue({
        body: '![The why not both girl](https://
   media3.giphy.com/media/3o85xIO33l7RlmLR4I/giphy.gif)'
      })
      // Creates a comment with a markdown image
      return context.github.issues.createComment(params)
    }
  })
}
```

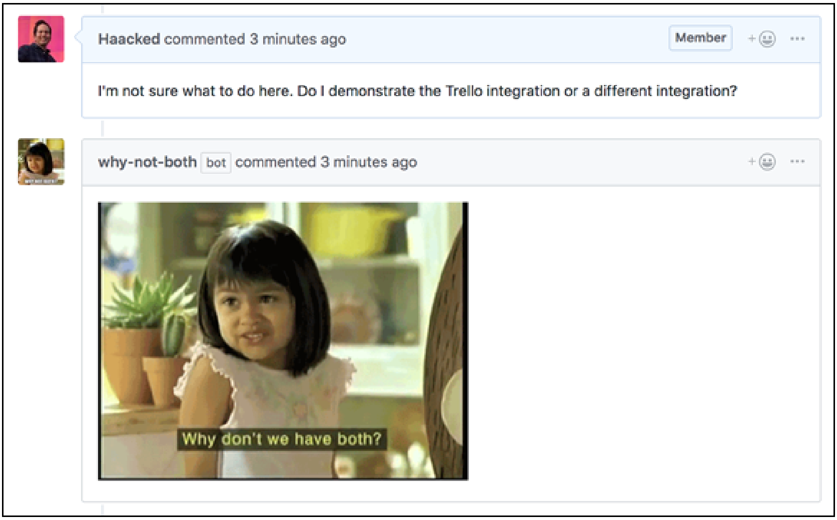
This code listens to new issue comments, looks for the word `or` surrounded by spaces, and if it finds it, creates a new comment with a markdown image.

This approach is not very smart. You can find a slightly better approach at `https://git.io/fhHST`. It would be even better if we could employ some artificial intelligence (AI) in the form of natural language processing (NLP). But that's beyond my skillset and out of the scope for this book.

Installing the app

After you create the app and get it working, others can install the app and use it. For example, we already built a version of the app that we are currently describing in this section. You can install it by going to `https://github.com/apps/why-not-both` and clicking the big green Install button in the top right.

Install it on a repository and then go create a comment on an issue in the repository that asks a question with the word `or` in it. An example interaction is shown in Figure 14-5.


Haacked commented 3 minutes ago

I'm not sure what to do here. Do I demonstrate the Trello integration or a different integration?

why-not-both bot commented 3 minutes ago

FIGURE 14-5: The why not both app in action.

Taking Action with GitHub Actions

In the previous section, we walk through personalizing GitHub by creating a GitHub app. This required that we host our app outside of GitHub. It turns out GitHub has a feature that removes the need to host our app outside of GitHub, which can reduce the number of moving parts when extending GitHub. This feature is called GitHub Actions.

GitHub Actions is one of the newer, most exciting features of GitHub. At the time of writing, it's still a beta feature. GitHub Actions makes it possible to create custom workflows on GitHub. It lets you implement custom logic to respond to events on GitHub. In the previous section, we wrote a GitHub app to do that. With GitHub Actions, we don't need to build a custom app. We can build workflows using existing actions that others have written, or we can write our own actions that run in a Docker container.

TIP Hopefully, by the time you read this, GitHub Actions are generally available. But in the case they're still in beta, email `support@github.com` and ask to be in the GitHub Actions beta program.

To demonstrate GitHub Actions, consider the following scenario. When you merge a pull request in your own repository, the branch for the pull request sticks around. GitHub presents a button to delete the branch, but a lot of people forget to do so and leave these branches sticking around.

Not deleting the branch isn't necessarily a bad thing, unless you're the type of person that likes things to be tidy and cannot stand having a branch that's no longer needed lingering around. If you are that type of person, you're in luck. Jesse Frazelle is also that type of person, and she wrote a GitHub Action you can use in your own workflows. Her blog post, "The Life of a GitHub Action" at `https://blog.jessfraz.com/post/the-life-of-a-github-action`, is a good read to understand more details about the life cycle of a GitHub action.

Creating a GitHub action workflow

If you've been accepted in the GitHub Actions beta program or it has become more widely available, you should see an extra tab at the top labeled Actions when you view a repository.

The following steps walk through the process for creating a GitHub action workflow for a repository.

1. **Click the Actions tab to see your existing workflows.**

   If you don't have any workflows yet, you see a big green button to Create a new workflow, as shown in Figure 14-6.

2. **Click the Create a new workflow button to bring up the workflow designer view.**

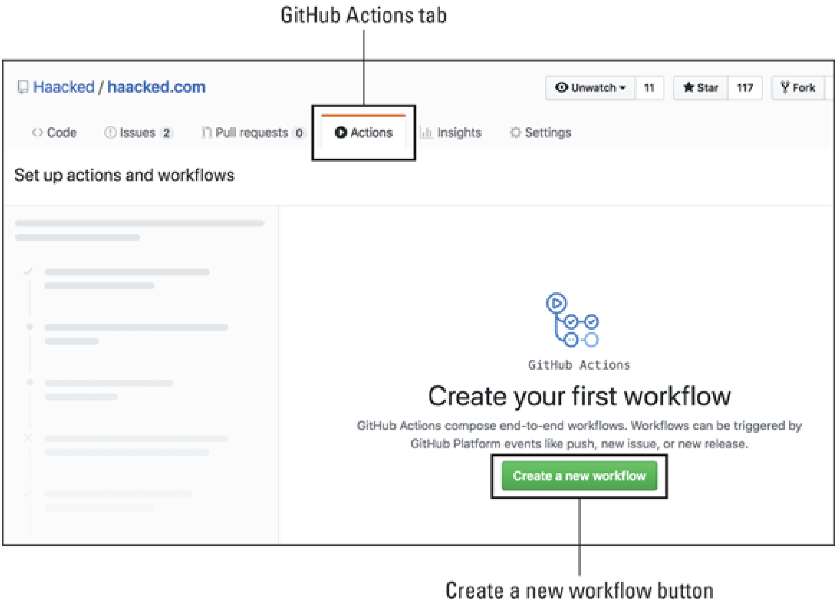
   This visual designer lets you create workflows and connect them to actions. Workflows are stored in a file named `main.workflow` in the `.github` directory of your repository. There's also an editor view if you prefer to build your workflows with text.

3. **Switch to the text editor and paste in the following text:**

```
workflow "on pull request merge, delete the
   branch" {
  on = "pull_request"
  resolves = ["branch cleanup"]
}

action "branch cleanup" {
  uses = "jessfraz/branch-cleanup-
   action@master"
  secrets = ["GITHUB_TOKEN"]
}
```

If you switch back to the visual designer, it should look something like Figure 14-7.

4. **Click the Start commit button to commit this new `main.workflow` file to your repository.**


GitHub Actions tab
Haacked / haacked.com
Unwatch
11
Star
117
Fork
Code
Issues 2
Pull requests 0
Actions
Insights
Settings
GitHub Actions
Create a new workflow
Create a new workflow button

Set up actions and workflows

Create your first workflow

GitHub Actions compose end-to-end workflows. Workflows can be triggered by GitHub Platform events like push, new issue, or new release.

FIGURE 14-6: The landing page for GitHub Actions workflows.

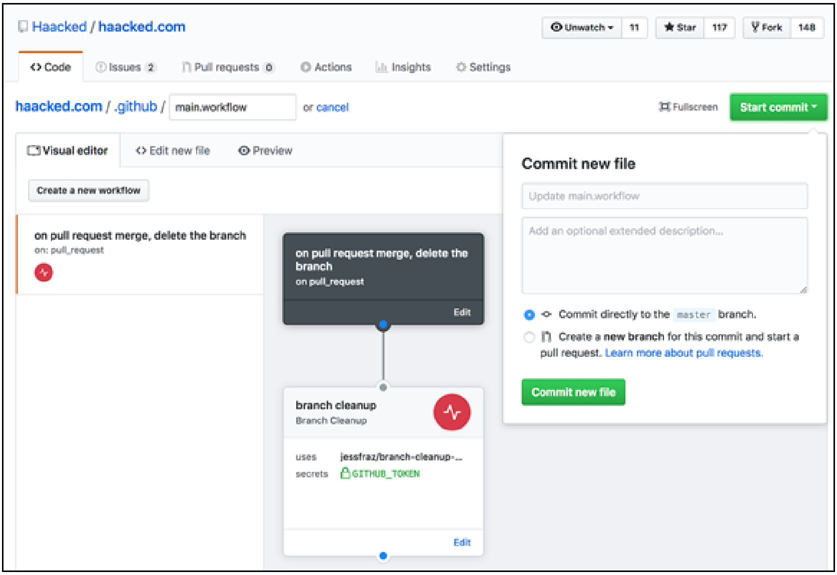

Haacked / haacked.com
Unwatch 11
Star 117
Fork 148
Code
Issues 2
Pull requests 0
Actions
Insights
Settings
haacked.com / .github / main.workflow or cancel
Fullscreen
Start commit
Visual editor
Edit new file
Preview
Create a new workflow
on pull request merge, delete the branch
on: pull_request
on pull request merge, delete the branch
on pull_request
Edit
branch cleanup
Branch Cleanup
uses jessfraz/branch-cleanup-...
secrets GITHUB_TOKEN
Edit
Commit new file
Update main.workflow
Add an optional extended description...
Commit directly to the master branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit new file

FIGURE 14-7: Committing a simple workflow.

Testing a GitHub Action

After you commit this file to your repository, the workflow is active. You can test it by creating a new pull request and then merging it. A few seconds or minutes later, you should see that the pull request branch was deleted. An update on the pull request says something like

```
github-actions bot deleted the branch-name branch 1 minute ago
```

TIP

You can create pretty useful and complex workflows using existing actions. You can install actions from the GitHub Marketplace or reference them by pointing to a repository that contains an action. You can also write custom actions. How to do that is beyond the scope of this book, but you can look at the action that Jesse wrote at `https://github.com/jessfraz/branch-cleanup-action` to get an idea of what it takes.

With GitHub Actions, you can personalize GitHub to your tastes in nearly unlimited ways.

Part 6

The GitHub Ecosystem

IN THIS PART …

Review the GitHub Marketplace for apps to enhance your project.

Install apps for continuous integration, code quality, localization, and more.

List your app on the GitHub Marketplace.

Explore the community on GitHub.com through people and repositories.

Find events where you can discover community beyond GitHub.com.

Discover strategies for speaking at your first event.

Chapter 15

Exploring the GitHub Marketplace

IN THIS CHAPTER

- **Introducing the GitHub Marketplace**
- **Finding apps in the Marketplace**
- **Installing apps from the Marketplace**

In the three chapters of Part 5, we look at multiple different ways of extending GitHub and customizing the GitHub experience. Many tools extend or integrate with GitHub. A good way to find tools to use with GitHub is the GitHub Marketplace.

Introducing the GitHub Marketplace

The GitHub Marketplace (`https://github.com/marketplace`) is a directory of tools and apps grouped in the following categories:

- Chat
- Code quality
- Code review
- Continuous integration
- Dependency management
- Deployment
- Learning

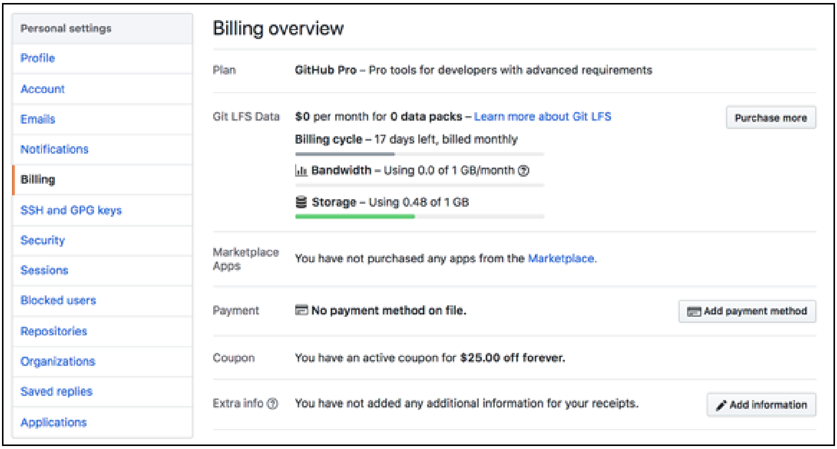
- Localization
- Mobile
- Monitoring
- Project management
- Publishing
- Recently added
- Security
- Support
- Testing
- Utilities

The Marketplace is a great way to find an app for every situation on GitHub. Purchasing or installing apps through the Marketplace has two key benefits: ease of billing and installation and the vetting process.

Billing made easy

For apps in the GitHub Marketplace that require payment, installing the app through the Marketplace is a streamlined flow because you can use your GitHub payment info. That way, you're not dealing with five different payment providers when purchasing five different apps to use with GitHub.

If you have a free GitHub account, you may not have setup your payment information in GitHub. To set up a payment method, click your avatar in the top right corner of GitHub.com and click Settings. From this page, click Billing from the list on the left side. Here you can click the Add payment method, as shown in Figure 15-1.


Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Sessions
Blocked users
Repositories
Organizations
Saved replies
Applications
Purchase more
Add payment method
Add information

Billing overview

Plan GitHub Pro – Pro tools for developers with advanced requirements

Git LFS Data $0 per month for 0 data packs – Learn more about Git LFS

Billing cycle – 17 days left, billed monthly

Bandwidth – Using 0.0 of 1 GB/month

Storage – Using 0.48 of 1 GB

Marketplace Apps You have not purchased any apps from the Marketplace.

Payment No payment method on file.

Coupon You have an active coupon for $25.00 off forever.

Extra info You have not added any additional information for your receipts.

FIGURE 15-1: Your Settings page on GitHub.com where you can add payment information.

The Marketplace vetting process

One of the benefits of installing an application from the Marketplace is that these apps must meet certain requirements before GitHub will list them in the Marketplace. The requirements help ensure a higher standard of quality and security with the apps; helping ensure that these apps are useful (no Fart apps) and are secure.

WARNING At the moment, a GitHub Action doesn't require any review to be listed in the GitHub Marketplace, which means installing an action from someone you don't know may be a bit riskier.

An app must meet four main categories of requirements before being listed in the Marketplace (`https://developer.github.com/marketplace/getting-started/requirements-for-listing-an-app-on-github-marketplace`):

- **User experience:** This brief set of nine requirements includes things like the app must have a certain number of users and installs already. It also includes some requirements around the behavior of the app, such as the app must include links to documentation, it can't actively persuade users away from GitHub, and it must provide value to customers.
- **Brand and listing:** This set of guidelines and recommendations

center around the branding of your app and your app’s listing. Every app must include its own logo. If the app makes use of GitHub’s logo, it must follow GitHub’s Logos and Usages guidelines. The brand and listing section on the Requirements page has links to further logo and description guidelines. As you can see, GitHub takes listing apps in the Marketplace seriously.

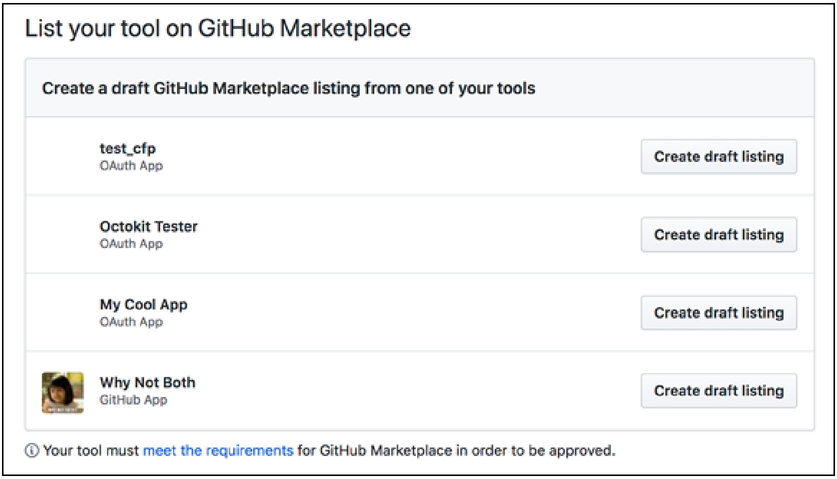
- **Security:** GitHub will conduct a security review of apps before listing them in the marketplace. A separate document with security best practices and more details on the security review is at `https://developer.github.com/marketplace/getting-started/security-review-process`.
- **Billing flows:** Every app in the Marketplace must integrate billing flows using the GitHub Marketplace webhook event. This requirement ensures that people can purchase a subscription to your app and cancel that subscription with the payment info they already have on file with GitHub. It also ensures that any changes made through GitHub are reflected immediately on the app’s own website.

Listing Your App on the Marketplace

Getting your own app listed in the Marketplace may increase the potential audience for your application. However, listing your app requires that it meets GitHub’s requirements and receives approval. Chapter 14 guides you through creating your own app.

To start the process of listing an app, click the Submit your tool for review link at the bottom of the Marketplace landing page or navigate to `https://github.com/marketplace/new` in your browser.

This page lists your applications that you can turn into Marketplace listings, as shown in Figure 15-2. Notice the Why Not Both app we create in Chapter 14 is listed here.



List your tool on GitHub Marketplace

Create a draft GitHub Marketplace listing from one of your tools

test_cfp
OAuth App

Octokit Tester
OAuth App

My Cool App
OAuth App

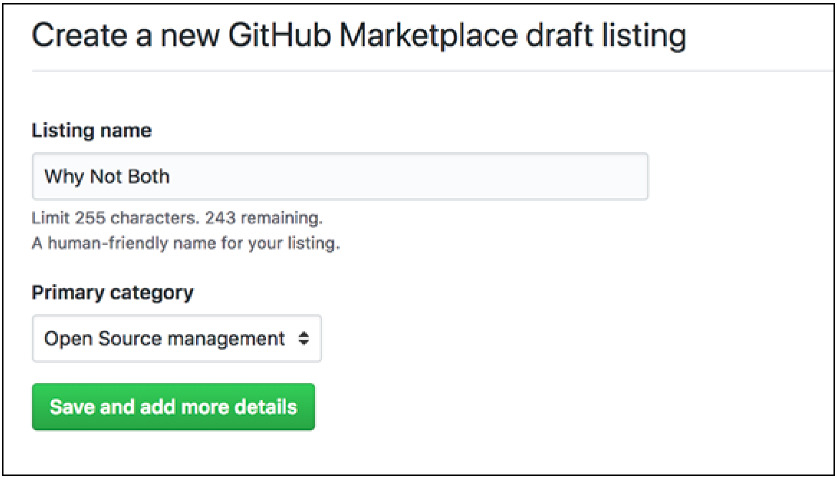
Why Not Both
GitHub App

ⓘ Your tool must meet the requirements for GitHub Marketplace in order to be approved.

Create draft listing
Create draft listing
Create draft listing
Create draft listing

FIGURE 15-2: Your applications that you can turn into Marketplace listings.

Click Create draft listing next to the app you want to list on the Marketplace to start the process. This takes you to a page where you can enter a name for the listing and choose one of the marketplace categories for your app listed earlier in the chapter, as shown in Figure 15-3.



Create a new GitHub Marketplace draft listing

Listing name: Why Not Both

Limit 255 characters. 243 remaining.
A human-friendly name for your listing.

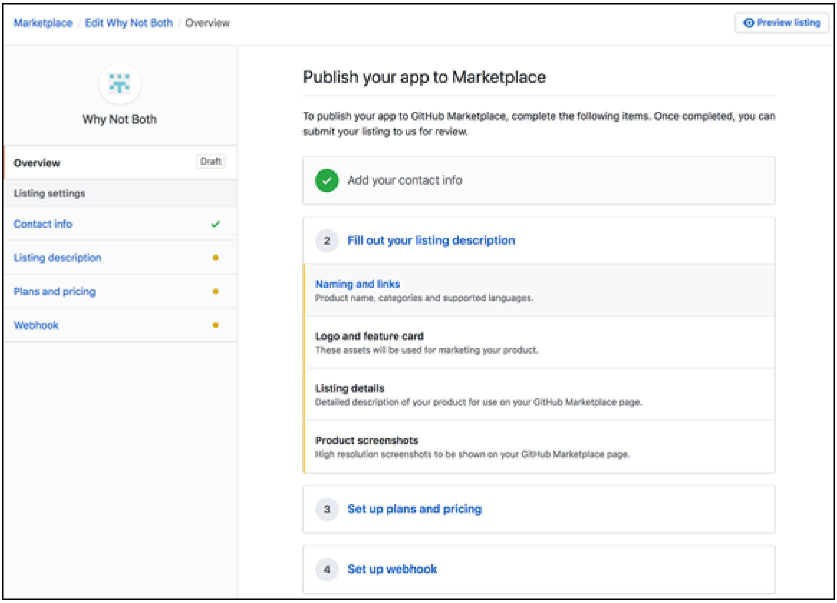
Primary category: Open Source management

Save and add more details

FIGURE 15-3: Filling out a form to list an app.

TIP If you save the draft of your listing but happen to close your browser, you can get back to your listing by going to `https://github.com/marketplace/manage` in your browser.

After you fill in these details, click Save and add more details to save a draft of your listing and move on to the next set of steps, as shown in Figure 15-4.


Marketplace / Edit Why Not Both / Overview
Preview listing
Why Not Both
Overview
Draft
Listing settings
Contact info ✓
Listing description ●
Plans and pricing ●
Webhook ●

Publish your app to Marketplace

To publish your app to GitHub Marketplace, complete the following items. Once completed, you can submit your listing to us for review.

✓ Add your contact info

2 Fill out your listing description

Naming and links
Product name, categories and supported languages.

Logo and feature card
These assets will be used for marketing your product.

Listing details
Detailed description of your product for use on your GitHub Marketplace page.

Product screenshots
High resolution screenshots to be shown on your GitHub Marketplace page.

3 Set up plans and pricing

4 Set up webhook

FIGURE 15-4: Steps to fill out a Marketplace submission.

These steps include

1. **Add your contact info.**

   This info is a set of three email addresses: Technical lead, marketing lead, and finance lead.

2. **Fill out your listing description.**

   This area is where you fill out more details, such as a product description, logo, and screenshots. The information here will be displayed on the Marketplace page for your application.

3. **Set up plans and pricing.**

   This is where you can set up one or more pricing plans, including the option to create a free plan, a monthly plan, or a monthly per user plan. You can also specify whether a plan includes a 14-day free trial.

4. **Set up webhook.**

   This step allows you to specify a URL where Marketplace events will be sent via an `HTTP POST` request. The webhook will send you information about events, such as purchases, cancellations, and changes like upgrades and downgrades.

5. **Accept the Marketplace Developer Agreement.**

   In order to list your app in the marketplace, you have to accept the Marketplace Developer Agreement.

6. **Click the Submit for review button.**

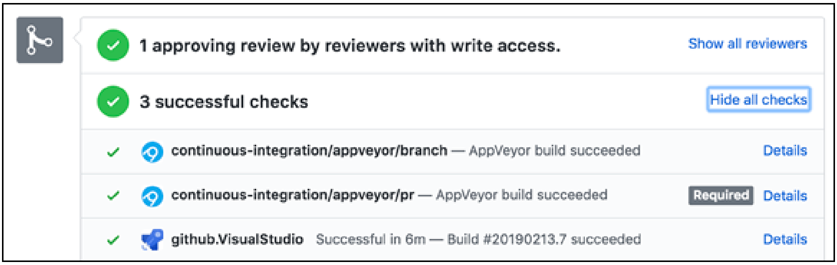
   GitHub employees will review your submission to make sure it meets the requirements to be listed in the Marketplace.

Considering Common Apps to Install

In the section "Introducing the GitHub Marketplace" at the beginning of this chapter, we list the categories of apps that are available on the Marketplace. In this section, we describe some of the most common and useful apps that you may want to consider installing.

Continuous integration

Continuous integration (CI) apps automatically build and test your code every time you push it to GitHub. If you have a CI app, such as AppVeyor, installed on your repository, you will see the status of the check at the bottom of each pull request, as shown in Figure 15-5.



1 approving review by reviewers with write access.

3 successful checks

✓ continuous-integration/appveyor/branch — AppVeyor build succeeded

✓ continuous-integration/appveyor/pr — AppVeyor build succeeded

✓ github.VisualStudio Successful in 6m — Build #20190213.7 succeeded

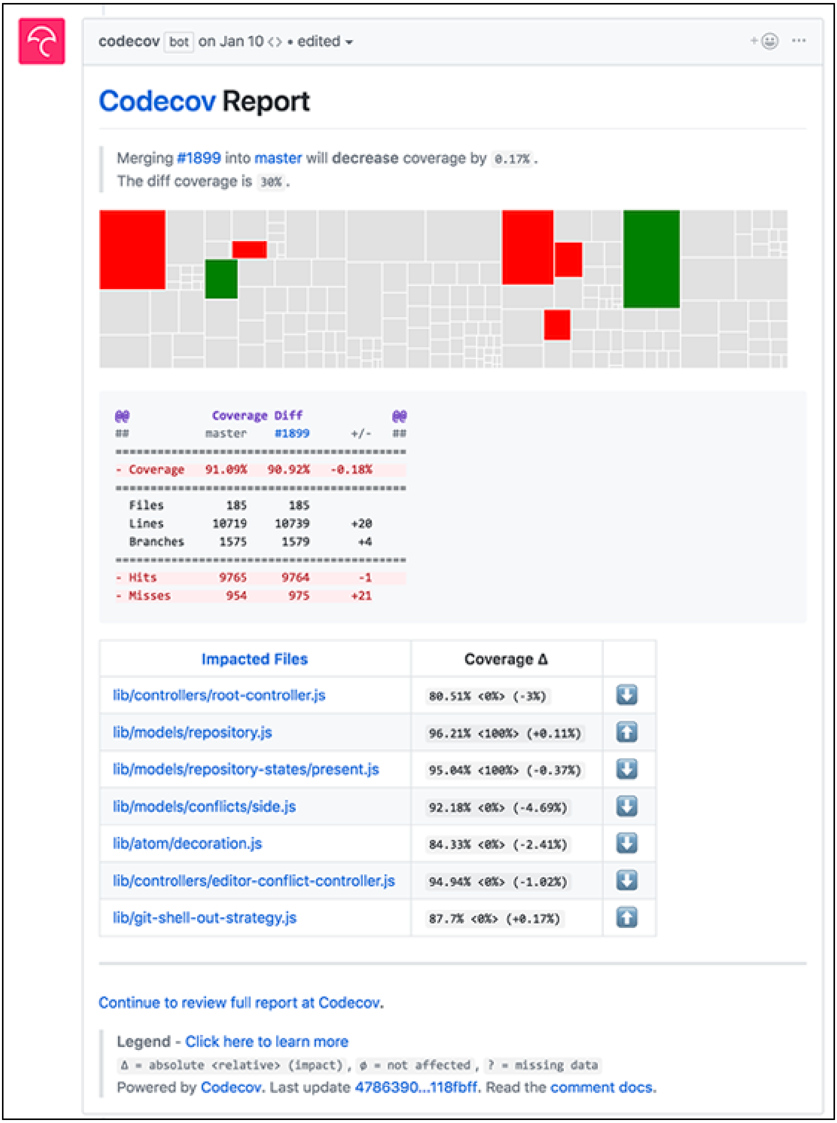
FIGURE 15-5: AppVeyor CI app example on the GitHub for Visual Studio repository.

If you're the owner of the repository, you can also specify whether checks have to pass before the branch can be merged into the master branch. Just head into the Settings tab. If you have any rules on the master branch already, click edit; otherwise, click Add Rule. From there, you can scroll down and select Require status checks to pass before merging.

Code quality

Code quality apps automatically review your code with style, quality, security, and test-coverages checks. These apps can be really useful for ensuring your code is kept to a high standard. With well-styled and quality code, you're less likely to introduce or miss bugs. For example, if you require that all curly braces are on new lines and indented with one tab per nested brace, you're likely to be able to spot when something is incorrect. For example, Rubocop checks the style of your Ruby code while it's building and provides you with style feedback, such as casing for method names.

Another useful type of code quality apps is the code coverage apps, such as Codecov. Shown in Figure 15-6, Codecov and apps like it comment on pull requests with how much of the code is covered by test scenarios, helping to ensure your code remains well tested.



codecov bot on Jan 10 • edited

Codecov Report

> Merging #1899 into master will decrease coverage by `0.17%`.
> The diff coverage is `30%`.

```
@@            Coverage Diff            @@
##           master   #1899     +/-   ##
==========================================
- Coverage   91.09%   90.92%   -0.18%     
==========================================
  Files         185      185              
  Lines       10719    10739      +20     
  Branches     1575     1579       +4     
==========================================
- Hits         9765     9764       -1     
- Misses        954      975      +21     
```

Impacted Files	Coverage Δ	
lib/controllers/root-controller.js	`80.51% <0%> (-3%)`	⬇
lib/models/repository.js	`96.21% <100%> (+0.11%)`	⬆
lib/models/repository-states/present.js	`95.04% <100%> (-0.37%)`	⬇
lib/models/conflicts/side.js	`92.18% <0%> (-4.69%)`	⬇
lib/atom/decoration.js	`84.33% <0%> (-2.41%)`	⬇
lib/controllers/editor-conflict-controller.js	`94.94% <0%> (-1.02%)`	⬇
lib/git-shell-out-strategy.js	`87.7% <0%> (+0.17%)`	⬆

Continue to review full report at Codecov.

> Legend - Click here to learn more
> `Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data`
> Powered by Codecov. Last update 4786390...118fbff. Read the comment docs.

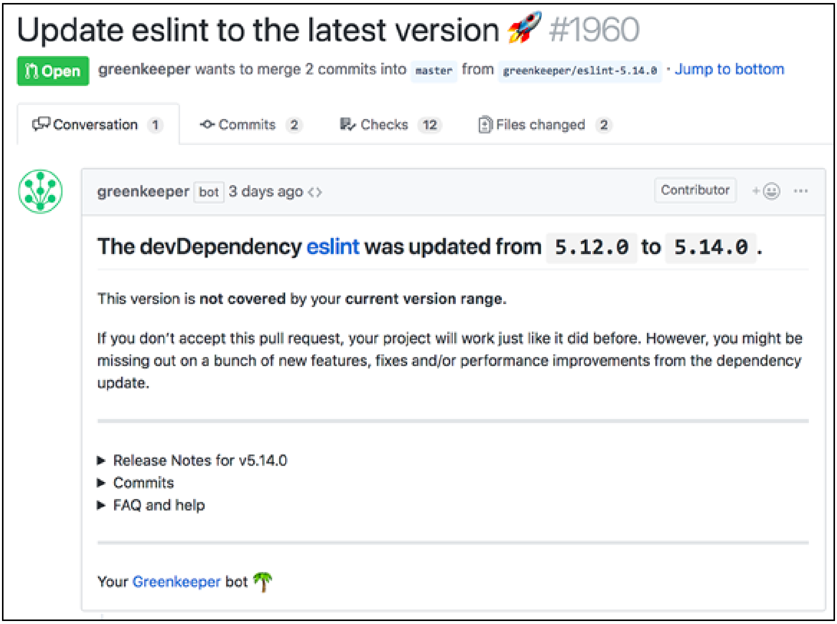
FIGURE 15-6: Codecov app example on the GitHub package for Atom repository.

Localization

Localization apps can make publishing your app in many languages easier. For example, the Crowdin app will link your repository to a Crowdin account where people from around the world can help you translate your documentation and any written words in your software (for example, on buttons or in menus). With more than 20,000 people contributing to translations, the Crowdin app will automatically open a pull request on your repository with new translations when it's reached a threshold of accuracy, still giving you a chance to review and merge. For open source projects, Crowdin is free!

Monitoring

Monitoring apps help measure performance, track errors, and track dependencies in your code. For example, Greenkeeper is a real-time notification app that gives you updates and changes for JavaScript dependencies. Figure 15-7 shows Greenkeeper in action, opening a pull request to update eslint to the latest version.



Update eslint to the latest version 🚀 #1960

greenkeeper wants to merge 2 commits into `master` from `greenkeeper/eslint-5.14.0`

greenkeeper bot 3 days ago

The devDependency eslint was updated from `5.12.0` to `5.14.0`.

This version is **not covered** by your **current version range**.

If you don't accept this pull request, your project will work just like it did before. However, you might be missing out on a bunch of new features, fixes and/or performance improvements from the dependency update.

- ▶ Release Notes for v5.14.0
- ▶ Commits
- ▶ FAQ and help

Your Greenkeeper bot 🌴

FIGURE 15-7: Greenkeeper app example on the GitHub package for Atom repository.

Dependency management

Modern app development today is heavily dependent on public package managers for pulling in and managing dependencies. A typical app may have dozens, if not hundreds, of dependencies. Tracking which of these dependencies are up-to-date can be difficult. Github apps like Dependabot check to make sure your dependencies are up-to-date and submit pull requests to update the ones that are not.

Sometimes you don't want all your dependencies on a public package registry. For example, if you work in an enterprise, you may have internal packages that should remain private. A private package registry tool, such as MyGet, is useful in this case. MyGet works with NuGet packages and lets you set up a policy where pushes to a particular branch will initiate a build and the branch will be deployed to a custom NuGet feed hosted on MyGet.

Testing

Testing software is an important part of the software development lifecycle. Good testers develop test plans to ensure that testers do a good job of testing each release. Managing test plans and keeping track of the status of test runs is an important part of quality assurance. The TestQuality app integrates with GitHub to helping developers and testers create, run, coordinate, and monitor software testing tasks.

Learning

TIP A great way to learn GitHub is to install the GitHub Learning Lab from the Marketplace. Installing Learning Lab install a bot that walks you through interactive lessons on how to use GitHub through a set of tasks that you complete. The lab is free and lets you take as many courses as you like at your own pace.

Chapter 16

GitHub and You

IN THIS CHAPTER

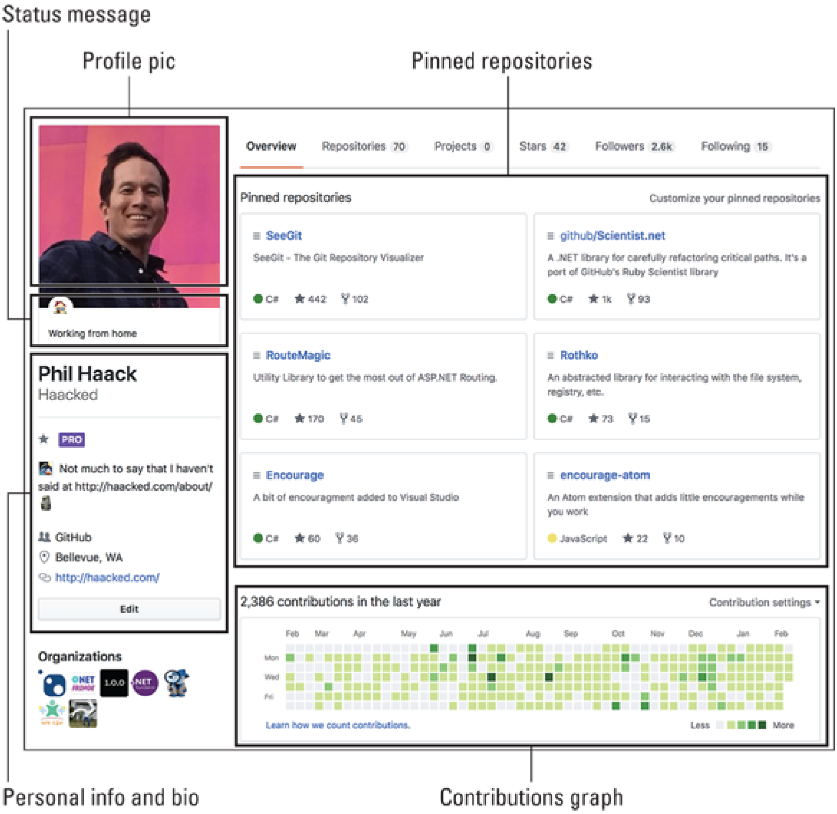
- **Making the most of your profile**
- **Understanding your contribution graph**
- **Following people and starring repositories**

GitHub is often described as a social network for developers. Throughout this book, we show you that GitHub is much more than a social network. It's an essential set of tools for working on code together. Even so, the social network aspect is still an important part of GitHub. It may well be a key reason for its success. When GitHub was created, there were existing source control hosts. Pretty much all these hosts were focused around projects. GitHub turned this project-focus approach on its head and made people the focus. You can see it in the URL structure where every repository has the name of the user or organization before the repository name.

In this chapter, we dig into the social network aspect of GitHub. We look at how you represent yourself in this network and how you can get involved in the online community.

Understanding Your GitHub Profile

Every user on GitHub has a profile page. Figure 16-1 shows the profile for one of the authors of this book at `https://github.com/haacked`.


Status message
Profile pic
Pinned repositories
Personal info and bio
Contributions graph
Overview
Repositories 70
Projects 0
Stars 42
Followers 2.6k
Following 15
Edit
Organizations
Contribution settings
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mon
Wed
Fri
Less
More

Working from home

Phil Haack
Haacked

PRO

Not much to say that I haven't said at http://haacked.com/about/

GitHub
Bellevue, WA
http://haacked.com/

Pinned repositories

Customize your pinned repositories

SeeGit
SeeGit - The Git Repository Visualizer
C# ★ 442 102

github/Scientist.net
A .NET library for carefully refactoring critical paths. It's a port of GitHub's Ruby Scientist library
C# ★ 1k 93

RouteMagic
Utility Library to get the most out of ASP.NET Routing.
C# ★ 170 45

Rothko
An abstracted library for interacting with the file system, registry, etc.
C# ★ 73 15

Encourage
A bit of encouragment added to Visual Studio
C# ★ 60 36

encourage-atom
An Atom extension that adds little encouragements while you work
JavaScript ★ 22 10

2,386 contributions in the last year

Learn how we count contributions.

FIGURE 16-1: The profile page for Haacked.

Your profile page represents you on GitHub. When you open an issue or submit a pull request to a new repository, the maintainers are likely to take a look at your profile to get a sense of you.

Not only that, your GitHub profile can serve as a portfolio of your development work. It provides some insight into your interests, experience, and ability as a software developer. Many companies who are hiring will take a look at your GitHub profile.

WARNING Many companies give heavy weight to applicants that have a GitHub profile. However, this biases against people who don't have the benefit of working at a company that uses GitHub. It also biases against those who don't have free time to work on open source. GitHub, Inc. itself doesn’t require a strong GitHub profile to apply for a job. More and more companies are taking an approach where they’ll look at your GitHub profile if you have one, but won’t hold it against you if you don’t.

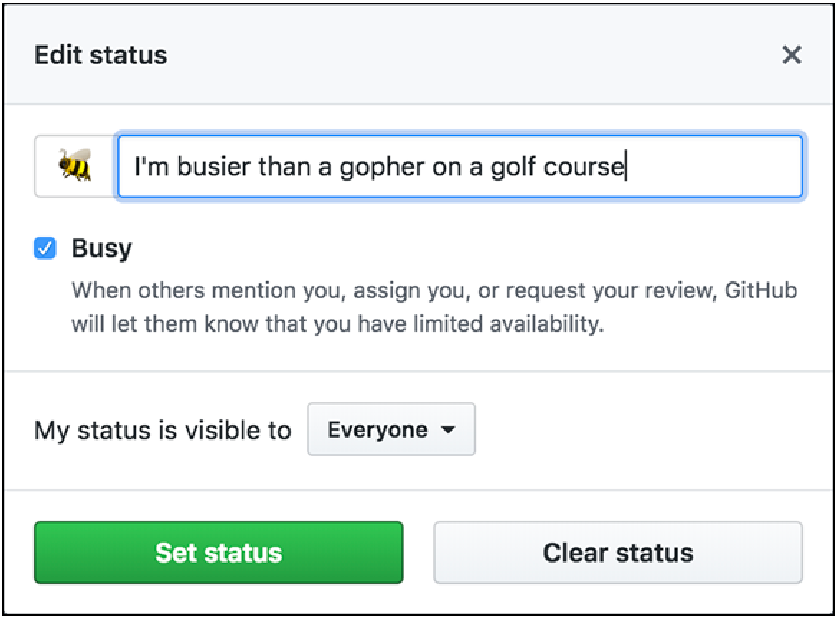
Profile picture

The first thing people visiting your profile will notice is your profile picture. This pic is associated with all your activity on GitHub. When you create an issue or a pull request or leave a comment, your profile pic is right there next to it.

TIP Your photo is an important part of your GitHub identity so make it reflect your personality, whether it’s a picture of you smiling warmly, a photo of a cartoon character, or a landscape photo.

Status message

Under your profile pic is a status message you can use to communicate something about yourself to the entire community. For some, it is an outlet to say something funny or meaningful. But for others, it’s used for practical purposes. For example, if you’re a maintainer of a popular project, you may want to let the world that you’re busy if you plan to be away from GitHub for a while. That sets the expectation that you may be slow to respond to new issues. Click your status message to bring up the option to change it. Figure 16-2 shows the status message dialog box with a busy message specified.



Edit status

I'm busier than a gopher on a golf course

☑ **Busy**

When others mention you, assign you, or request your review, GitHub will let them know that you have limited availability.

My status is visible to Everyone

FIGURE 16-2: Changing your status message.

Personal info and Bio

GitHub displays your bio and information that you choose to show the world under the status message. Click the Edit button to change your bio, company name, location, and URL. This area is a good opportunity to tell the world a bit more about yourself and link to your personal blog or website.

REMEMBER All this information is completely optional. By entering it, you give GitHub permission to display this information wherever your user profile appears. You can delete the information at any time by editing your profile.

In this section, you also sometimes find badges that GitHub will add to users' profile. For example, in Figure 15-1, you can see that Haacked has the Pro badge. Haacked is definitely a pro GitHub user, but that's not what that badge means. A Pro badge on a GitHub profile just means that that user pays for the individual Pro subscription. (You can see different pricing models at `https://github.com/pricing`.) Another badge you may often see is the Staff badge. This badge is reserved for full-time employees at GitHub (also called *Hubbers*). Figure 16-3 shows meaghanlewis's profile with the staff badge. Meaghan happens to be the technical reviewer for this book and is an engineering manager at GitHub!

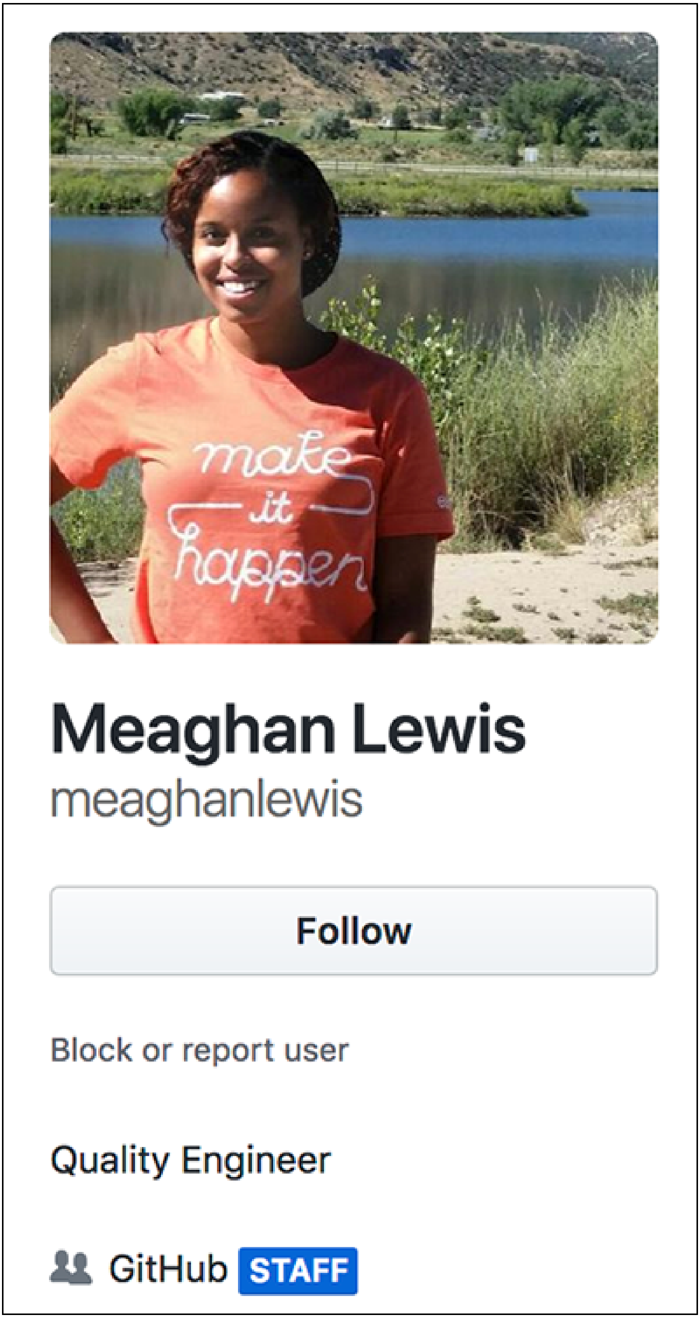

Meaghan Lewis
meaghanlewis
Follow
Block or report user
Quality Engineer
GitHub STAFF

FIGURE 16-3: Staff badge on a GitHub profile.

Pinned repositories

By default, GitHub shows a selection of your most popular repositories on your profile page, but those repositories may not represent what's important to you. Click Customize your pinned repositories to select up to six repositories to pin to your profile page.

Pinned repositories can be useful from three perspectives:

- **You:** Pinned repositories are useful to you because they're a quick link to the repositories you care about most. Pinning repositories can make it quick and easy for you to get there, instead of having to create bookmarks for each of them.
- **Other GitHub users:** When other people visit your profile on GitHub, the first introduction to the kind of work you do is the set of your pinned repositories. This will give folks a sense of what you're interested in. It also highlights the areas where you have experience. Listing areas of experience can be important if you're just starting to contribute to a new open source project because maintainers will often visit your profile page to get a sense of who you are.
- **Companies:** Hiring managers and recruiters may look at your profile when you apply to a job to get a sense of your experience. Pinning some projects that you have most contributed to or are most proud of to the top of your profile can help them to get an accurate picture of you.

REMEMBER

You can also always change up your pinned repositories based on what you're doing. For example, if you're applying for a new position, you may want to refresh what is pinned to be more specific to the role you're applying to. And if you're trying to learn something new, you may want to pin repositories that you're currently engaged with. Think of your entire profile page as a living document — one that should be updated as your goals and interests change.

Contribution graph

The *contribution graph* is a grid of squares 7 squares high and 52 squares in length representing each day of the past year. Each square is filled in with a color that represents your contribution level on that day. If you didn’t make any contributions, the square remains gray. If you made some contributions, the square ranges from light green to dark green, depending on how many contributions you made.

Issues and pull requests count as contributions on standalone repositories. Commits to a standalone repository’s default branch (typically `master`) or to its GitHub pages branch (typically `gh-pages`) count toward your contribution graph.

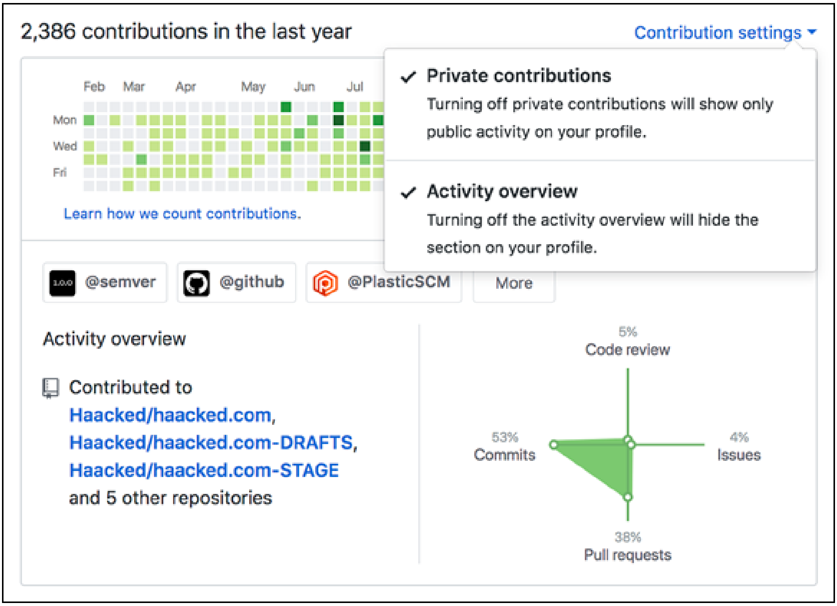
TIP If you fork a repository, an Issues tab won't appear at the top of the repo because, typically, the goal of a forked repository is to contribute back to the original open source project. You fork the repository, make your changes, and open a pull request that targets the original repository. If you really want to have issues in a forked repository, you can turn them on in the Settings tab, though we don’t recommend it as you should be tracking your progress on an issue in the original repository. Because the intended behavior of a fork is to contribute back to the original repository, issues on the forked repository and pull requests that do not target the original repository do not count toward your contribution graph. The thought here is that you haven’t yet contributed to the main project, the fork is kind of just like your own private branch and the true contribution is made after you’ve added thoughts to an issue or merged code in on the original repository.

The contribution graph is one of GitHub’s more controversial features. Two common concerns are raised. The first is that it promotes unhealthy behavior in that many people attempt to keep long streaks of activity going. Many people take pride in having activity on every single day of their graph, even weekends. Though you may be working on something that makes you happy on the weekends, which is okay, it’s also very important to recognize that it’s not — and shouldn’t be — expected that you are coding every single day of your life. Some of the most important aha! moments have come from taking a break and gaining a new perspective.

The other concern is that other people draw bad conclusions about a person’s ability as a developer based on the activity graph. For example, someone may look at a developer’s contribution graph, see very little activity, and conclude that he’s not very productive.

WARNING Drawing any conclusions from other people’s contribution graphs doesn’t make sense. A contribution graph should be useful only to yourself as a fun way to see your history of activity. For one thing, the contribution graph is easily gamed. In fact, a repository provides code for doing pixel art using your contribution graph at `https://github.com/nikhilweee/github-activity-art`. It’d be easy to use that tool to make your contribution graph completely dark green. The contribution graph isn’t meant to be used as a definitive productivity metric.

Also, contributions to private repositories may not be showing up in a contribution graph. Click Contribution settings to change that setting. Figure 16-4 shows an example of both enabling the display of private contributions as well as an activity graph.



2,386 contributions in the last year

Contribution settings ▾

✓ **Private contributions**

Turning off private contributions will show only public activity on your profile.

✓ **Activity overview**

Turning off the activity overview will hide the section on your profile.

Learn how we count contributions.

Activity overview

Contributed to **Haacked/haacked.com**, **Haacked/haacked.com-DRAFTS**, **Haacked/haacked.com-STAGE** and 5 other repositories

FIGURE 16-4: Displaying private contributions and the activity graph.

Unlike the contribution graph, which shows how much activity you have, the activity graph shows where your activity occurs. As you can see in Figure 16-4, one of the authors of this book has most of his activity in commits and pull requests.

Contribution activity

Underneath the contribution graph (not shown in Figure 16-1) is the contribution activity timeline. This timeline of your activity on GitHub goes all the way back to your first commit. It can be nostalgic to go back to the beginning of your activity.

WARNING In some cases, you may have activity in your timeline that shows up before you joined GitHub. You may even have activity that occurred before GitHub was created! How is that possible? The activity timeline includes Git activity in your repositories based on the timestamp of the Git commits. It's possible to import a repository into GitHub that was created before GitHub existed, which would cause you to have activity prior to GitHub's creation.

Starring Repositories

TIP When you visit a useful repository, you can star it by clicking the star in the top right corner of the repository page. A star is not only a compliment to the repository owner, but also serves as a bookmark of sorts for the repository.

On your profile page, click the Stars tab to see a list of all the repositories that you've starred. This list is viewable by others who happen upon your profile page. Exploring the repositories others have starred is a great way of discovering interesting new projects.

TIP If you're curious about which repositories have the most stars on GitHub, use GitHub search (`https://git.io/fhdkx`) to sort users by follower count. This shortened URL shows the top starred repositories on GitHub.

TIP You may be curious about which of your repositories have the most stars. Right now, you can't list your repositories in order of stars. However, one of the author's visualization website at `https://profile-summary-for-github.com/user/haacked` provides a nice visualization of your profile and includes which of your repositories have the most stars (see Figure 16-5). Just replace the username *haacked* with your own to see your profile.

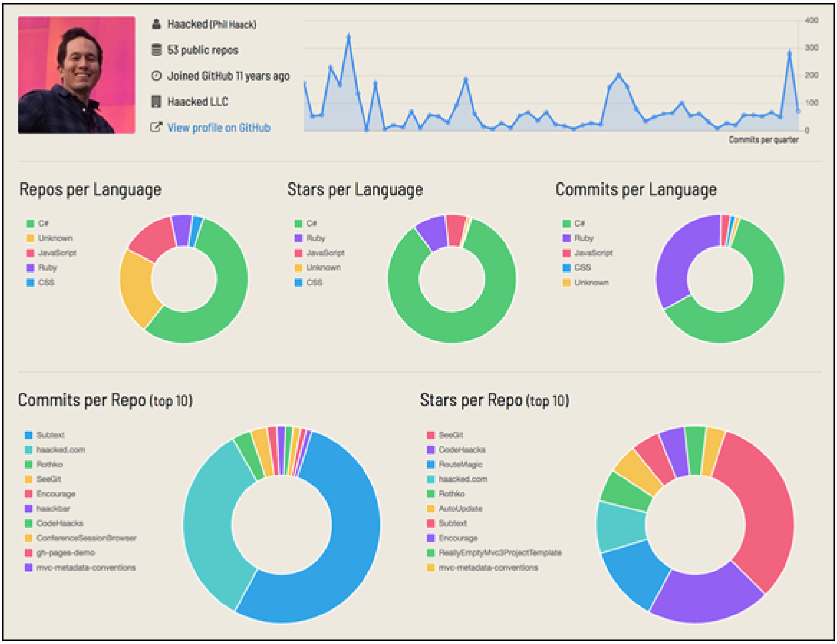

Haacked (Phil Haack)
53 public repos
Joined GitHub 11 years ago
Haacked LLC
View profile on GitHub
400
300
200
100
0
Commits per quarter
Repos per Language
C#
Unknown
JavaScript
Ruby
CSS
Stars per Language
C#
Ruby
JavaScript
Unknown
CSS
Commits per Language
C#
Ruby
JavaScript
CSS
Unknown
Commits per Repo (top 10)
Subtext
haacked.com
Rothko
SeeGit
Encourage
haackbar
CodeHaacks
ConferenceSessionBrowser
gh-pages-demo
mvc-metadata-conventions
Stars per Repo (top 10)
SeeGit
CodeHaacks
RouteMagic
haacked.com
Rothko
AutoUpdate
Subtext
Encourage
ReallyEmptyMvc3ProjectTemplate
mvc-metadata-conventions

FIGURE 16-5: Profile visualization for one of the authors.

Following Users

When you visit the GitHub profile of another user, a big Follow button appears underneath the profile picture. Click the Follow button to subscribe to notifications about the user's activity in your dashboard at `https://github.com`. Who you follow also feeds into GitHub's recommendation system. For example, if someone you follow stars a public repository, that repository may show up in the Discover repositories section of your dashboard as a recommendation.

You can see all the users you follow by clicking the Following tab of your home page. You can see all the people that follow you by clicking the Followers tab.

TIP If you're curious about who are the most followed people on GitHub, you can use GitHub search (`https://git.io/fh7P7`) to sort users by follower count. This shortened URL shows the top followed people on GitHub. It may come as no surprise that Linus Torvalds, the creator of Linux and Git, is the most followed person on GitHub.

Chapter 17

Attending Events

IN THIS CHAPTER

- **Growing your career through events**
- **Finding an event in your area**
- **Speaking at events**

Starting a career as a software developer is very challenging, especially if you do it on your own. This is one reason why GitHub is such an essential tool for software developers. As the biggest source code host in the world, it's also the biggest software developer community. By participating in GitHub, you become connected to the world's experts and best teachers for any technology you may be interested in.

Most people who use GitHub barely scratch the surface of what GitHub offers. By reading this book, you're a step ahead of many other developers. The knowledge of how to use GitHub will serve you well. But writing code on GitHub only scratches the surface of a rewarding career as a developer.

In this chapter, we look at how to get two steps ahead in your career by encouraging you to step away from the keyboard for a moment and meet with other developers in-person. Attending events is a great way make connections that will benefit you during your entire career. And speaking at events is a great way to grow in your career even more. It may feel daunting, but everyone has something to offer others, even those just starting out.

Exploring Types of Events

Many kinds of events focus on software developers. They range from the informal meet-up or user group to the structured multiday international software conference. In this section, we cover the most common types of events and what to expect at each.

Meet-ups and user groups

A *meet-up* or *user group* is an informal gathering of developers to cover a topic. Many are scheduled monthly and hosted by a local company or software interest group.

These events tend to be a great way to dip your toe into software developer events. They tend to be small gatherings of people in your area. Each month features a local speaker who talks about a topic relevant to the group. (Some user groups and meet-ups bring in speakers from outside on occasion, but typically they focus on highlighting local speakers.)

TIP Meetups.com is a great way to find a meet-up relevant to your interests. For a list of JavaScript meet-ups in Seattle Washington, go to `https://www.meetup.com/topics/javascript/us/wa/seattle`. You can search for meet-ups by location on meetups.com.

A few examples of local meet-ups include

- **Brooklyn JS:** `http://brooklynjs.com` in Brooklyn, New York
- **.NET São Paulo:** `www.meetup.com/dotnet-Sao-Paulo` in São Paulo, Brazil
- **SD Ruby:** `www.sdruby.org` in San Diego, California

Regional conferences

A *regional conference* is a relatively small conference where speakers and attendees outside of the local area are welcome, but the focus of the conference is to provide a venue for local developers and speakers to connect and present their work.

Often these conferences are one or two days. Many will have a single track of talks, or two at most. They're a step up in size and structure from a meet-up and typically occur once a year, as opposed to monthly.

Some of them often offer workshops either before or after the conference. These workshops usually cost extra, but offer more in-depth training for a specific skillset or technology. For example, you can often find a full-day workshop dedicated to improving your Git skills. If you can afford it and find one that teaches a skill you want to improve, workshops are often worth the investment.

Some great examples of local conferences include

- **.NET Fringe:** `http://dotnetfringe.org` in Portland, Oregon
- **JSConf Hawaii:** `www.jsconfhi.com` in Honolulu, Hawaii
- **GoRuCo:** `http://goruco.com` in New York, New York

Hackathons

A *hackathon* is very different from a conference. While conferences focus more on having speakers teach a topic through a talk, hackathons focus on building. A hackathon is an event that may last several days where groups of people form teams to work together collaboratively write code to solve some sort of problem.

The usual format is some sort of problem is presented and teams are tasked with building a solution. The technology stack they may use is often dependent on the focus of a hackathon. For example, a mobile development hackathon will require that attendees build a mobile app to solve the problem.

Hackathon is a portmanteau of the words hack and marathon. Many take the marathon aspect to the extreme by having teams work around the clock with very little sleep. Others try to create a balance of working hours and sleeping hours by forcing contestants to leave the workspace.

TIP Hackathons are often very inclusive of beginners. For example, `www.sdhacks.io` is open to any high school or college student in the world who is 18 or older. You don't always have to have a team when you sign up for a hackathon. Often, you can find one when you get there. It's best to check out the FAQ for the specific hackathon to learn more about the details.

One of the largest, worldwide hackathons is targeted to college students. It's the Microsoft Imagine Cup (`https://imaginecup.microsoft.com/en-us/Events?id=0`). Winners of the Imagine Cup can win mentorship from Satya Nadella (Microsoft CEO), travel to the world championship, and receive Azure grants and $100,000.

Attending hackathons can be a great way to be introduced to a new technology. The goal isn't to design and implement a final product, but rather to hack together bits and pieces to make progress on an idea that you have. The end product should look more like a prototype than a polished application. Often times, hackathons will have mentors that know a particular technology that you can learn from. Think of a hackathon as a dedicated time and place to experiment and learn.

TIP Though attending a formal hackathon will provide you with mentors, a space, and sometimes prizes, you can also always get together with friends to do one on your own, too! Just pick a time, place, and goal and try to hack together a prototype of an idea you have! It doesn't hurt to give it a shot!

Major conferences

A major conference tends to be large and draw attendees from all over the country, if not the world. Attendee counts tend to be in the thousands. Attending one of these conferences requires a bit more up-front planning. It's not just arranging your flight and hotel. These conferences tend to have many tracks, so for any given time slot, you may have to choose which talk you want to see from five or more talks at the same time.

Like a regional conference, major conferences often offer an array of workshop offerings before or after the conference. In addition to workshops, many also include hands on labs during the conference. Labs are usually included in the price of the conference and offer a great chance to actually try out the technologies you're hearing about at the conference.

Many of these conferences are thrown by large technology companies, such as Microsoft's Build conference and Apple's WWDC.

A few examples include

- **Oscon:** `https://conferences.oreilly.com/oscon/oscon-or` in Portland, Oregon

- **Build:** `www.microsoft.com/en-us/build` (location changes each year)
- **WWDC:** `https://developer.apple.com/wwdc in` San Francisco, California

Knowing What to Expect at Events

Events can vary widely in terms of what to expect, but they all have a few commonalities. The most obvious thing to expect is that other developers will be there. Not everyone has the benefit of living in a tech hub. If you live outside of a tech hub, being a developer can feel solitary. If you work at a company that is not primarily focused on software, it can feel lonely. A software event is an opportunity to meet likeminded individuals — people who really care about the craft of software and improving themselves. Events are a good chance to make connections.

Keynotes

Many conferences, especially the larger ones, will include a keynote talk. Some include more than one. A keynote talk sets the tone for a conference and is usually related to the theme of a conference. For a multitrack conference, no other talks are usually scheduled during the keynote.

TIP For a major conference held by a large software company such as Microsoft or Apple, the keynote is where they'll often make major announcements of new products and features.

Conference session tracks

The primary draw of conferences are the session tracks. A *track* is a set of talks, typically organized around a theme. Smaller conferences may have only a single track, while larger conferences may have a large number of tracks. A user group meeting or a meet-up may only have a single talk.

Depending on a conference, a session can range from 30 minutes to 75 minutes. Many of them end with some time for audience questions and answers (Q&A).

WARNING If you participate in a Q&A, it's considered rude to simply use that time to make a statement. Make sure that your question actually ends with a question mark.

Sometimes presenters will ask to follow up with you after the talk to have a more in-depth conversation. Don't assume that it's because they don't want to answer in a public forum. Typically, Q&A lasts only 5 to 15 minutes, and there isn't always enough time to fully understand a question and formulate an effective answer. You can meet up with presenters directly after the talk (though you should let them at least get their laptop and things off the stage) or ask when they may be able to grab a cup of coffee during the conference if it's a multiday conference. You can also ask whether there is an asynchronous way to follow up with them that they would prefer.

REMEMBER Conference presenters are typically attending the conference to learn something as well; they aren't only there to present, so be respectful of their time as well.

Hallway tracks

At most conferences, the sessions are very valuable if it's a topic you want to learn about, but just as valuable is what's known as the *hallway track*. The hallway track isn't one you typically find on the conference schedule. It's a term that people coined to refer to the informal conversations you have in the hallway in between sessions, during lunch, or when you skip a session.

Sometimes you'll come across a world expert in a topic having a casual conversation with a few attendees, and you'll be welcomed to join in. These hallway tracks can often be even more educational than attending a talk because you can engage in a conversation instead of just listening. They also may lead to lifelong friendships and interesting collaborations.

TIP Meeting new people at a conference can be intimidating, especially if you're an introvert. It may help to know that most people feel this way. If you draw up the courage to introduce yourself to someone you don't know who is alone, she may feel relieved. If you're an extrovert, look for opportunities to draw people who are alone into a conversation in a low-pressure manner.

After-hour conference events

Many conferences include after-hour events. Some of these events can be quite lavish depending on the size of the conference.

WARNING Often, these events include loud music and alcohol, so be aware if that's not your thing.

Many conferences try to be more creative and inclusive with their attendee events. For example, one conference rented out a table tennis place with a large number of tables.

A respectful professional environment

At most events, you can expect a respectful, inclusive, and professional environment. This environment is conducive to networking, socializing, and learning.

There have been cases where conferences failed to live up to that expectation. Because of that, many conferences now adopt and enforce a code of conduct. A *code of conduct* outlines behavioral expectations of the participants in a conference. More importantly, it communicates to folks who are often the targets of harassment that the conference takes harassment seriously and is not welcome at the event.

Becoming Familiar with GitHub Events

Given that this book is about GitHub, we'd be remiss not to include some information about GitHub's own events. GitHub hosts and sponsors several conferences throughout the year.

To see which of these events are upcoming, you can visit `https://github.com/events`. This page lists GitHub's own upcoming events, as well as events that it sponsors.

GitHub Universe

GitHub Universe (`https://githubuniverse.com/`) is the flagship conference for GitHub. It is held yearly in the city where GitHub's main headquarters resides, San Francisco, California. The conference is usually held in the fall around October or November.

As GitHub describes it,

GitHub Universe is a conference for the builders, planners, and leaders defining the future of software.

This conference is where GitHub typically makes its biggest announcements of the year during the keynotes. It attracts well-known speakers from prominent software companies.

In 2018, the conference had around 1,800 attendees who attended three tracks of talks. The cost of the conference is reasonable, around $99 per person, or $199 if you also attend the workshops.

GitHub Satellite

The GitHub Satellite conferences (`https://githubsatellite.com/`) are an offshoot of GitHub Universe. They bring a GitHub universe-style conference to locations around the world.

Held once a year, past Satellites have been held in places such as Berlin, Tokyo, and London.

GitHub Constellation

GitHub Constellation (`https://githubconstellation.com/`) is a series of small community events held multiple times a year around the world. These events focus on the local community and often feature speakers local to the area. They are typically free and occur over one or two evenings. They're not all-day conferences like Satellite and Universe.

Git Merge

Git Merge (`https://git-merge.com/`), is a conference sponsored by GitHub, but focused on the Git version control tool and the people who use it every day. As GitHub puts it,

Through technical sessions and hands-on workshops, developers and teams of all experience levels will find new ways to use, build on, and scale Git.

The conference features a preconference hands-on day of workshops focused on a range of Git topics. This conference is a great one to learn more about Git and to improve your Git skills.

Speaking at Events

A lot of developers have the misconception that speaking at conferences is only for experts who have been in the industry for years or that only big extroverted personalities speak at conferences. We're here to tell you this is not true.

Everyone has a story to tell

Whether you realize it, somewhere inside of you, you have an interesting story to tell — even if you're relatively new as a software developer. One thing all these events have in common is they're better off when they have a diverse set of speakers with diverse viewpoints. For example, it's common for experienced developers to lose their "beginner's mind" when working with a technology for so long. A talk by a beginner about their struggle to learn a particular technology is often eye-opening and just the kick in the pants experienced developers need to make it better.

Benefits of being a speaker

Being a speaker offers a lot of benefits. The main one is that it's a forcing function to spend time deeply learning a topic. If you plan to give a talk on a subject, it's a good idea to research it beyond what you already know. And teaching a topic to others is a great way to solidify your own understanding.

Not only that, it's a great way to receive feedback on your ideas. Often, after giving a talk, someone will have a unique insight to share that improves upon your ideas. You wouldn't have received that feedback without putting your ideas out there.

Another benefit is the exposure and networking opportunities that being a speaker entails. Because your badge will have the word

SPEAKER on it, people are more likely to want to meet you and talk to you. When you've spoken at a few conferences, it gets your name out there. When you apply for a new job, people may give you more opportunities because they've heard you speak.

And a few perks come with being a speaker. Often, conferences have special speaker-only events where speakers can get to know each other. This event often leads to a great networking opportunities and friendships because it's easier to remember people in a smaller setting. Also, as you speak at more and more conferences, you may see some speakers at multiple conferences and even become friends. It helps when you go to a conference to already know some of the other people there because you've spoken together before.

Finding Funding for Events

Some of these events and conferences can get expensive. Microsoft BUILD can cost upwards of $2,500, and Google I/O is close to $1,200 — not to mention the transportation and parking to get to the conference and the hotel and flight costs if you don't live near the venue.

Many software companies will pay to have employees attend a conference if the conference is a valuable learning opportunity. If you work for a software company, it doesn't hurt to ask. To make your case stronger, explain how the things you learn at the conference will improve your performance at work.

Larger software companies may also be sponsoring events and need volunteers to attend and help run the booth. Some companies even offer a stipend for attending conferences as a perk and part of your compensation package.

Another way to fund your trip is to apply for a scholarship. The Grace Hopper Celebration is a large conference with more than 18,000 attendees and celebrates women in computing. With technical tracks as well as tracks focused on diversity and inclusion, this conference is typically held over three days and moves around the United States for the venue. The Anita Borg institute that puts on the conference also offers scholarships that you can find at `https://ghc.anitab.org/2019-student-academic/scholarships`. These scholarships typically include airfare, hotel, transportation costs, meals, and a ticket to attend. They focus on students and faculty for this particular scholarship.

Most conferences will waive the price of the conference ticket for speakers. Many conferences will also cover hotel and travel to the conference. And some conferences will even offer an honorarium on top of expenses. These benefits completely depend on the conference. Furthermore, some conferences ask for volunteers to help run the conference, which also tends to come with some perks like a free ticket so that you can enjoy the rest of the conference when you're not working one of your shifts. The Grace Hopper Celebration calls it the Hopper Program, which you can find at `https://ghc.anitab.org/hoppers`.

REMEMBER If you've found an event you're particularly excited to attend, before you shell out the thousands of dollars from your own pocket, ask around! Ask your network if anyone knows of any scholarships, apply to be a speaker or volunteer, and ask your company or school what resources are available to you!

Part 7

The Parts of Tens

IN THIS PART …

Find resources on GitHub.com to support your growth.

Discover how to keep up-to-date with GitHub Integrations.

Explore external resources for learning more about collaborative coding.

Understand techniques to improve your workflow on GitHub.com.

Find tools that will help improve your workflow.

Become an effective community member with guided tips.

Chapter 18

Ten Ways to Level Up on GitHub

IN THIS CHAPTER

- **Finding resources on GitHub.com**
- **Exploring online resources in the broader web**
- **Keeping informed about GitHub integrations**

Becoming an expert on GitHub is not a quick task. First, you need to master a lot of specific features of GitHub. Knowing how to create a pull request or link issues to a project board with automation for effective project management is one aspect of this expertise.

It is also important to begin to master specific areas in software engineering as a whole to be able to effectively contribute to projects in meaningful ways. Furthermore, becoming an effective community member is another way to become a GitHub expert.

This chapter briefly describes ten ways you can level up in each of these areas so that you can be successful on GitHub.

Trial and Error

TIP

One the best ways to learn anything is to just try. When you try, you always learn. If you try and fail, you learn what not to do and gain insight into how something works. When you succeed, you learn what to do next time! Learning how to use

GitHub and how to code is no different. In fact, learning by trial and error in the tech field is even easier.

TIP

GitHub provides you with unlimited public and private repositories, which means you can try all kinds of things without ever spending a dollar! We highly recommend getting onto GitHub.com and creating a new repository. From there, make a `README.md` file that you can easily modify and work with. This is the “Hello World” app of a GitHub repo. You can create issues and pull requests that modify the `README.md` file just to see how they work.

To learn more about collaboration on GitHub, invite your friends to be collaborators on your repo or make your repo public and send them the link to it. Ask them to comment on issues and create and review pull requests. Ask your friends to make their own public and private repo where you're a collaborator so that you can try all the GitHub features from the perspective of a contributor. There is no harm in trying things out.

After you finish exploring, you can always delete the repositories, if you want. keeping some around may be useful for the inevitable time when GitHub releases new features that you want to try on a dummy repo.

GitHub Help Docs

The GitHub help docs are extensive and detailed. You can find them by going to `https://help.github.com`. The help docs can be extremely useful if you know exactly what you want to get done. With 38 categories that each have anywhere from 1 to 70 docs, hundreds of pages describe every single feature of GitHub with images and cross referenced links.

At the top of the GitHub Help page (where all the docs are located) is a search bar where you can type a query, and docs related to that query will appear. Similar to Google, as you start typing your query, results will start to appear. If you press Enter or click the magnifying glass button, a page with search results appears. On this page is a notation next to the doc title with the category that it is a part of. We recommend paying attention to that notation because it will help you better navigate docs in the future. If you understand how GitHub categorizes the docs, you may be able to find what you're looking for faster next time.

If you click a specific doc, you see the documentation page. On the right side of this page is a list of Article versions. The documentation defaults to GitHub.com documentation, but GitHub also offers a slightly different version to enterprise users. Clicking on the different article versions displays the docs specific to that version of GitHub, ensuring that you have the most accurate information.

TIP Extremely security conscious companies are more likely to use GitHub Enterprise, as it allows them to have the interface and functionality of GitHub while keeping their code and data secure on their own servers. This ability can be important if you have proprietary code that you don't want anyone, not even GitHub, to have access to. If you're planning on open sourcing your code (which a lot of large companies, like Microsoft, do a lot of) or you just want your code to be private from the world, storing on GitHub.com is typically good enough.

At the bottom of each documentation page you will typically find a section called Further reading. This section lists documentation pages that may be relevant to what you're trying to do. Below this section is a button that says Contact a human. If the documentation isn't helping you resolve your issue, click this button to go to a page where you can contact GitHub. The Contact page has an extensive list of all the ways you can try to help yourself first, but then provides a quick and easy message section for you to ask a question. On the left side of the contact page you can also find other reasons for contacting GitHub, such as reporting abuse, reporting content that isn't appropriate, reporting copyright infringement, privacy concerns, and portals for both premium and enterprise support, if you've purchased those plans. You can also directly access this page at `https://github.com/contact`.

WARNING This Contact page is not where you should submit a question or concern about a specific repository unless you're reporting abuse, content, or copyright infringement. If you have a problem getting code to run or have a question about a specific repository, you should ask in an issue directly on that repository home page on GitHub.

At the top of the GitHub Help page is a drop-down list where you can select which version of GitHub you're interested in looking at the documentation for. This selection defaults all the docs to that version so that you don't have to change it every time you visit a page. There is also a quick link to the Contact page and a quick link to return to the GitHub home page.

REMEMBER Docs are your friend. There is a lot to learn and software changes quickly; expert programmers are experts not just because they know a lot, but mostly because they know how to figure something out — and that often means knowing how to find information.

An alternative to the docs is the GitHub Guides found at `https://guides.github.com`. These visual, quick five to ten minute guides explain a specific action you can take in GitHub — for example, how to fork a project. These guides are great reminders for how to do something and provide a bit more insight than the documentation page for a specific feature.

GitHub Learning Labs

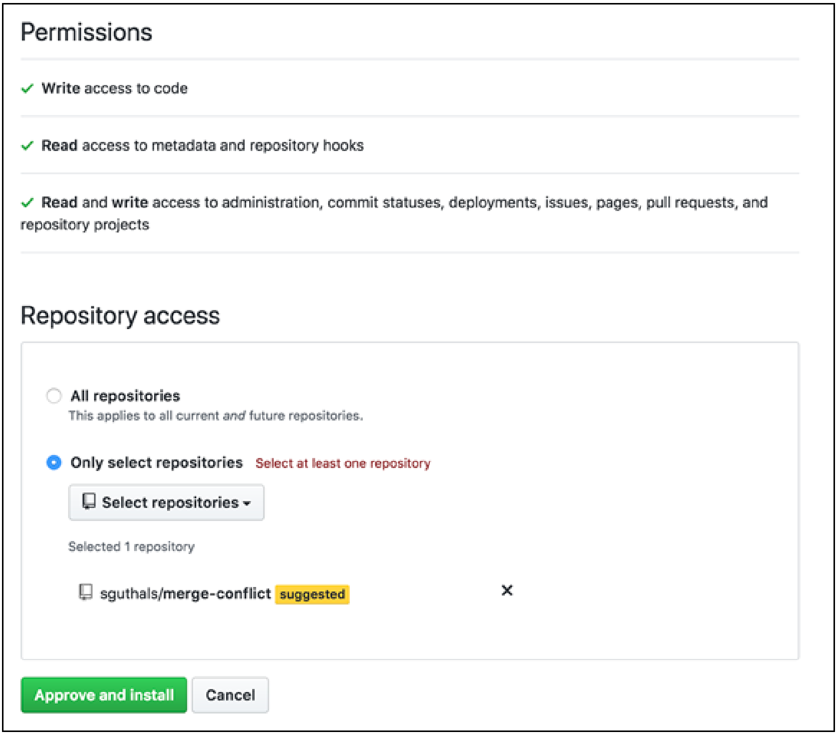
GitHub has invested in a team of folks dedicated to helping novices learn how to use GitHub. And though documentation can help when you know exactly what you're looking for, sometimes it can be hard to follow still images. That's why GitHub brings you Learning Labs, which are absolutely incredible. Learning Labs are step-by-step, guided tutorials where you actually create a real repository on GitHub and perform real actions that you would normally. The friendly GitHub

Learning Lab Bot guides you through the tutorial, creating pull requests for you to review or issues for you to close out.

To get started with the Learning Labs, go to `https://lab.github.com` and click the blue Sign in with GitHub button. Now you can browse a number of courses that may be useful for you to get an in-depth introduction into everything GitHub.

On the Courses page, at `https://lab.github.com`courses, you can find 14 (maybe more by the time you read this book) courses categorized into topics such as Git, GitHub, and Open Source. With some of the most commonly used features on GitHub being highlighted, you can learn to review pull requests, manage merge conflicts, and make your open source project stand out among the millions of projects on GitHub.com.

Clicking on a course takes you to a detailed page where you can learn more about that course and join it. Often times, the Learning Lab will want to create a public repository on your account. Don't worry; you can always delete it after you've finished the course, but there is also nothing wrong with showing the world that you take learning seriously. You will also be asked to accept the Terms and Conditions and to install the Learning Labs app onto your account. This app installs only on the repos you select, so you can have it install on the repo that it creates as part of the course. For example, doing the merge conflict course, you may give it access only to the merge-conflict repo that it creates, as shown in Figure 18-1.


Permissions

✓ Write access to code

✓ Read access to metadata and repository hooks

✓ Read and write access to administration, commit statuses, deployments, issues, pages, pull requests, and repository projects

Repository access

○ All repositories
This applies to all current *and* future repositories.

● Only select repositories Select at least one repository

Selected 1 repository

sguthals/merge-conflict suggested

FIGURE 18-1: Giving the GitHub Learning Lab app access to a specific repository.

After you install the Learning Lab app, return to the Learning Lab course page (for example, `https://lab.github.com/install?org=githubtraining&course=managing-merge-conflicts`) and click the Done installed? Move on link under the green Install button. You're taken to a new page with each of the course steps. The course overview page is useful because you can always stop in the middle of the course and pick it up later if you aren't able to finish in one sitting.

A typical Learning Lab will have the bot open an issue in the repo that it created with a description of what you should do. All you have to do is read through the issue and follow the instructions!

TIP We recommend opening the repo in a new tab so that you can refer back to the instructions without losing your place in what you're trying to accomplish.

When you're just starting to get introduced to GitHub, Learning Labs is a great way to experiment with features and learn specific workflows. It's essentially trial and error, but with a helpful guide.

GitHub In-Person Training

GitHub is tool for more than 30 million developers and over 2 million organizations. From novice coders just learning to create their first GitHub Pages website in Markdown to Microsoft bringing you VS Code, GitHub's goal is to support more people building more software. As such, GitHub offers trainings where a GitHub expert will help better equip your team to use GitHub.

With eight different focus areas, plus a customized training in case you need something different for your team, GitHub will not only guide you through the details on how to use GitHub, but will engage everyone in the fundamental workflows and techniques to using it effectively. The focus areas are

- Workflow consultation
- Implementation
- Admin mentoring
- Train-the-Trainer
- Migration
- API consultation
- InnerSource
- Services account engineering

For a full description of each of these areas, as well as what you can get with a customized course, visit `https://services.github.com`. This service is not free. You can submit an inquiry and a GitHub representative will reach out with more information about costs.

If you're a novice, you may not need the full-force of a GitHub trainer. However, it may be something for you to suggest to your company if it's asking you to switch over to using GitHub. In-person training can also be useful if you start a company and want everyone on your team to not only use the tool, but to use it effectively.

Project-Specific Documentation

One of the best ways to learn about a specific open source community and a specific technology is to reach the project-specific documentation. When you are both new to GitHub and new to a technology, it can be effective to read through the documentation most relevant to your interest and to explore the behaviors of other members of the community by reading through issues and pull requests.

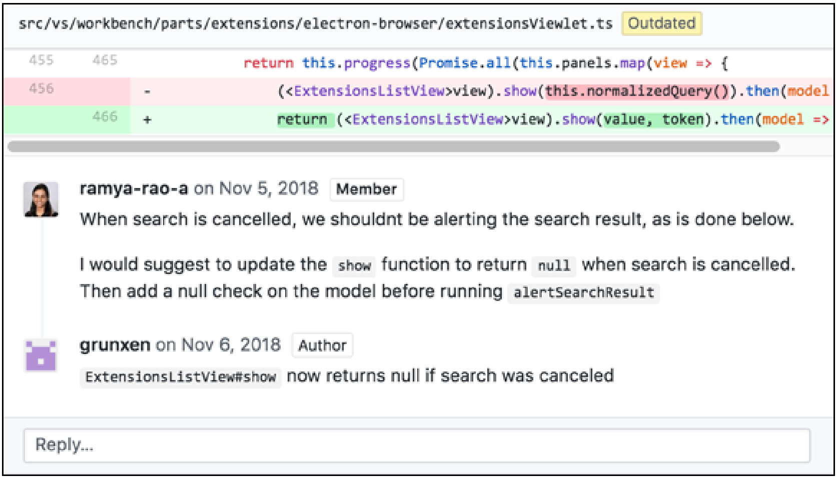
The VS Code project is a good example of a project with good documentation and enough community engagement to understand behaviors. If you go to `https://github.com/microsoft/vscode`, you can find a well-written `README` with links to where you can submit bugs and feature requests, how to contribute to the project, and where to engage with the core developers.

When you're first starting to learn, following the How to Contribute documentation (for example, VS Code has a wiki page for it at `https://github.com/Microsoft/vscode/wiki/How-to-Contribute`) can help you get your code up and running on your local machine. Even if you don't end up contributing directly to the project, following this documentation can be a good learning exercise.

REMEMBER Some projects are strict on their style and contributing guidelines. For example, the VS Code project has guidelines on style found at `https://github.com/Microsoft/vscode/wiki/Coding-Guidelines`. Having code that is consistent in style means that errors are less likely to occur. For example, if you always use PascalCase for type names and camelCase for function names, you can quickly identify when someone accidentally referred to a type instead of a function in their code.

It is also important to know the requirements for contributing code to the open source project. For example, on VS Code, you must first sign a Contributor License Agreement before you open a pull request on the repository.

Looking through open and closed pull requests can also be good practice when you're trying to learn about the specific community. Understanding what kinds of errors are common in the particular code base or what kinds of changes the core team typically requests can help you learn about the technology as well as the specific project. Successful open source projects will have pull request reviews with substantial information. For example, on one of the pull requests for the VS Code project, you can see user ramya-rao-a not only let user grunxen know that alerts shouldn't happen, but also provides a suggestion on how to fix this bit of code, as shown in Figure 18-2.



src/vs/workbench/parts/extensions/electron-browser/extensionsViewlet.ts

```
            return this.progress(Promise.all(this.panels.map(view => {
-               (<ExtensionsListView>view).show(this.normalizedQuery()).then(model
+               return (<ExtensionsListView>view).show(value, token).then(model =>
```

ramya-rao-a on Nov 5, 2018

When search is cancelled, we shouldnt be alerting the search result, as is done below.

I would suggest to update the `show` function to return `null` when search is cancelled. Then add a null check on the model before running `alertSearchResult`

grunxen on Nov 6, 2018

`ExtensionsListView#show` now returns null if search was canceled

FIGURE 18-2: Effective pull request review feedback on the VS Code open source project.

External Community Places

Some open source projects have places other than GitHub where more general conversations can happen. These places can be a really great way to learn without having to explicitly open an issue or pull request. You can find these extended communities many places online, some with a more interactive community while others are more feedback-driven.

The VS Code project, for example, calls out the `StackOverflow` questions that are tagged with visual-studio-code. If you have questions about VS Code, it's best that you first search the questions already asked. If yours is unique, make sure you use the appropriate tag when writing it.

This project also suggests giving feedback on Twitter, asking questions or directly giving feedback to the `@code` alias. Giving feedback is also a good way to get updates on the project.

The Atom project, on the other hand, has a specific forum run on its website found at `https://discuss.atom.io`. This forum is a great place to search for answers to your question or ask a unique one, if it hasn't been asked. As you become more expert in using Atom, you can start contributing answers.

Atom also has a public Slack that you can join. Just visit `http://atom-slack.herokuapp.com` and enter your email address. You will be sent an invitation to join the public Slack channel where you can ask questions and generally follow the discussion around Atom, Electron, or just general chat "around the water cooler". This resource is great if you're learning on your own. A Slack channel can be a virtual office where you have colleagues and peers that you can ask questions to in a lightweight manner (for example, not open an issue on GitHub).

Online Coding Tutorials

Depending on what you're trying to learn and contribute to on GitHub, an online coding tutorial may be right for you. Now, you can probably find thousands of coding tutorials online from a random person posting a blog to a YouTube video showing you exactly how to do something, but some of our favorite coding tutorials offer you an in-browser sandbox where you actually get to try out the code it's trying to teach you about.

When dealing with GitHub, you may be inclined to learn a bit of Ruby. If you ever want to create a GitHub app or integration, it may behoove you to know a bit about Ruby to make your life easier because GitHub is built using Ruby. The online tutorial `https://ruby.github.io/TryRuby` gives you a step-by-step guide with an editor and an output window right in your browser.

WARNING While the Ruby tutorial gives you a handy Copy button so that you don't have to type all the example code every time, it can be useful to go through the motions of typing to slow down and better learn what you're reading. We recommend typing the code, even if it seems simple enough, so that you get more practice.

If you're making a web application, you may want to get some practice with JavaScript, HTML, and CSS. `https://jsfiddle.net` is a great place to start, giving you an editor for each language along with an output window. It has also been enhanced to have boilerplate starter code for commonly used languages, such as React, TypeScript, and CoffeeScript.

Another interactive sandbox tutorial website that we particularly like is the Try .NET site at `https://dotnet.microsoft.com/platform/try-dotnet`. Before diving in to the intricacies of the .NET Framework, you can get your hands dirty in a safe environment right in your web browser. With snippets that you can try, tutorials you can follow, or simply an open playground to experiment, this site can offer you support at any stage in your career.

While you can definitely find other sites where you can code in the browser, these are just a few of our favorites. When you're first starting to learn a new language or framework, we recommend you explore various resources or ask for recommendations from peers or communities you're interested in joining before just following the first tutorial you find.

Online Courses and Tutorials

Online courses are becoming more popular with the development of better tools for engaging with learners. Two of the largest platforms for online courses are Coursera and EdX. These platforms partner with universities and large companies to provide in-depth education and sometimes certification or even degrees.

Both Coursera and EdX offer specializations with multiple courses. For example, there is a Ruby on Rails Web Development Specialization on Coursera that was created by Johns Hopkins University. Meanwhile, Microsoft offers a professional certificate in Introduction to Computer Science, which has three courses.

Furthermore, some universities offer degrees, such as the University of London and its bachelor of science in computer science degree on Coursera or University of California, San Diego's MicroMasters in data science on EdX. These programs often require an application and admission into the specific program — not just anyone can join.

For more lightweight options, sites like Udemy and Pluralsight offer more than 100,000 online courses at low prices or sometimes even free. These courses tend to be more focused on video lectures and tutorials rather than extensive coursework, but they're really great options for when you're first getting started and just want to get an introduction into something.

Khan Academy is another great place to find always free courses. Khan Academy supports learners as young as two on its Khan Academy Kids app (`https://www.khanacademy.org/kids?from=lohp`) to adults looking to learn something new from computing to studying for the LSAT to entrepreneurship and growth mindset! This great resource offers curated courses that are more lightweight than Coursera or EdX.

It is also important to not forget the power of YouTube. Though videos on YouTube aren't curated and aren't held to any standards, you can find a lot of really amazing content. For example, The LearnCode.academy channel has a GitHub Tutorial for Beginners video with almost two million views. Siraj Raval is also a great channel to follow as he gives tutorials, but also explains the history of certain technologies and how they actually affect our world.

Blogs and Twitter

If you're looking for a updates on certain technologies or products or a quick answer to something, it can often be fruitful to follow the right people on Twitter and subscribe to the right blogs. We won't be able to tell you who the right people are for you to follow, because each person reading this book may have different interests when it comes to learning and contributing on GitHub, but we can give you some tips!

First, recognize that people who use Twitter and who blog are usually only representing their own ideas and are not representing a company or being held to any company standards. This means that you should take what they say with a grain of salt and recognize that they are just people, saying what is on their mind, and probably also still learning and developing (as we all are always learning and developing).

When searching for Twitter accounts to follow, you can start with accounts representing the technology you're interested in learning about. For example, following `@GitHub` may be a good idea because this account will not only post updates to GitHub, but will also often highlight other tech, people, or events related that may be of interest to you. This will start to give you ideas on other people to follow. For example, GitHub recently retweeted Safia Abdalla (`@captainsafia`) who is a developer at Microsoft and is also an open source developer. She does keynotes and talks on open source development, too, so if you're interested in learning more about open source development, she may be a good resource of information!

Advice on which blogs to follow is similar to the advice on Twitter. We'd recommend starting with some of the blogs that may have the most relevant information — for example, the GitHub blog at `https://github.blog` gives you up-to-date information about GitHub features. Another great resource is Medium.com, where you can search any keyword and get a number of different perspectives and pieces on it. For example, `https://medium.com/search?q=github` results in a lot of lists of things you can do on GitHub that you may not have known about. It can be a great start to then looking into the GitHub docs for more in-depth information. Finally, you can also just Google a keyword, and various blogs, tutorials, and videos are likely to pop up.

Community Forum

We also recommend the GitHub Community Forum. especially for getting help about GitHub. Found at `https://github.community`, this forum tends to have hundreds of people actively online at any given time. The GitHub community is worldwide, and people from all different technological backgrounds are typically eager and willing to help.

Many threads contain topics specific to GitHub, such as how to use Git and GitHub, GitHub Pages, and the GitHub API. But this community forum offers even more than GitHub-specific information. As the number one place for developers, GitHub also provides a space for the community to learn from each other in general programming concepts and projects overall.

TIP Before participating in the GitHub Community Forum, we recommend reading through the Code of Conduct, found at `https://github.community/t5/Welcome/Code-of-Conduct/m-p/65#M37`. Just like an open source project, the community forum is an open place where one must remain respectful and maintain a collaborative spirit.click

From this community forum, you can also find the GitHub Education community forum at `https://education.github.community` focused on how to use GitHub in the classroom. This resource is great if you're a student and you'd like to convince your teacher to use GitHub or if you're a teacher looking to teach your students.

The GitHub Community Forum is a great place to ask your generic questions that wouldn't be appropriate to ask on a specific repository. The GitHub community is vast as it is a large portion of the developer community. Almost anywhere you go in the developer community, you are sure to find someone who can help you or at least point you in the right direction. The most important thing for you to remember is to ask questions, be respectful, and remember to give back with your knowledge as you begin to learn.

Chapter 19

Ten Ways to Improve Your Development Workflow

IN THIS CHAPTER

- **Discovering software tools that improve your development workflow**
- **Finding out how StackOverflow helps get your questions answered**
- **Realizing that some workflow improvements are techniques, not tools**

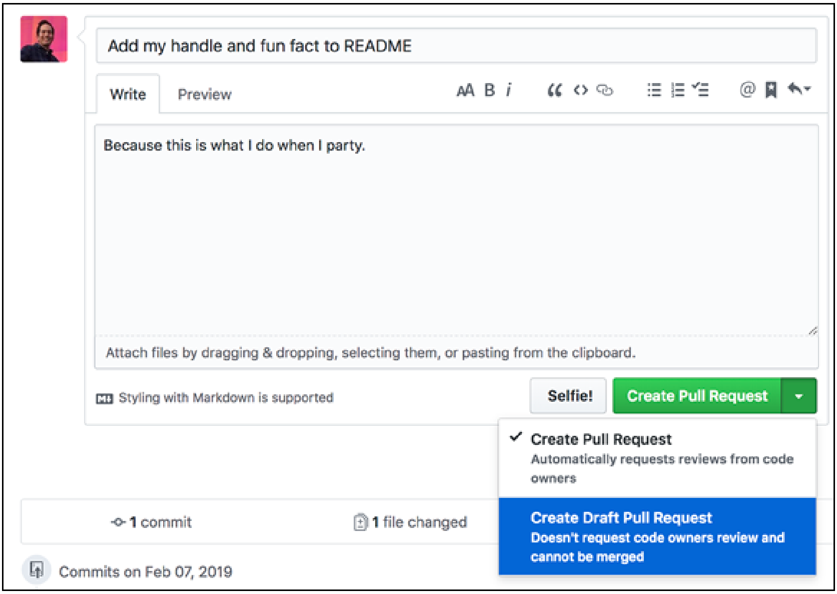
Working on software is tedious at times. Writing code is laborious and requires a ton of steps and intense concentration. On top of that, a lot of tasks have to occur during the coding process, such as running tests, creating mock-ups, and tracking progress.

Any tools and techniques you can use to help streamline and improve your development workflow not only saves you time, but can improve the quality of your work. It's always worth spending time periodically looking at ways to improve your development workflow. In this chapter, we cover ten ways you can improve your development workflow.

Drafting Pull Requests

Chapter 8 discusses creating pull requests when you're ready to have code reviewed and merged into the main branch of a repository. But that's not the only way to use pull requests. In fact, GitHub employees have long stated that creating a pull request is the beginning of a collaborative conversation. Sometimes it may be appropriate to create a pull request even before you've pushed code. It's possible to do by creating a branch on GitHub and then creating a pull request from that empty branch.

Or maybe you do have some code to push, but you know it's incomplete. You just want to gather some feedback on your progress without alerting code reviewers that they should give your pull request your full attention with a detailed code review. This is where draft pull requests come in handy. To draft a pull request, click the down arrow on the Create Pull Request button and then select Create Draft Pull Request, as shown in Figure 19-1.


Add my handle and fun fact to README

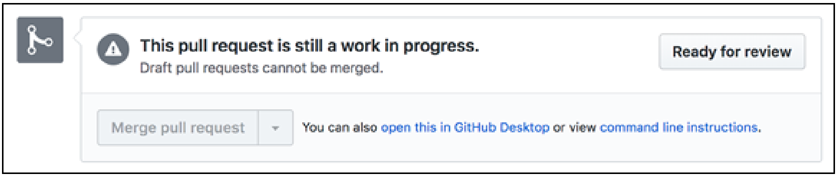
Because this is what I do when I party.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Write
Preview
Styling with Markdown is supported
Selfie!
Create Pull Request
✓ Create Pull Request
Automatically requests reviews from code owners
Create Draft Pull Request
Doesn't request code owners review and cannot be merged
1 commit
1 file changed
Commits on Feb 07, 2019

FIGURE 19-1: Drafting a pull request.

This creates a pull request in draft mode. If you have a `CODEOWNERS` file (which we cover in Chapter 11) in your repository, a draft pull request will not notify those reviewers until the pull request is marked as ready for review by clicking on the Ready for review button, as shown in Figure 19-2.


This pull request is still a work in progress.
Draft pull requests cannot be merged.

You can also open this in GitHub Desktop or view command line instructions.

Ready for review
Merge pull request

FIGURE 19-2: The ready for review button.

Git Aliases

Chapter 6 introduces the concept of Git aliases. Git aliases are short shell scripts that extend Git and automate common tasks. Chapter 6 includes a Git alias for moving commits from one branch to another.

Git aliases can automate tedious clean-up tasks. For example, the following alias deletes all branches that have already been merged into the target branch. If no target branch is specified, then it assumes the `master` branch.

```
[alias]
    bclean = "!f() { git branch --merged ${1-master} | grep -v
   " ${1-master}$" | xargs git branch -d; }; f"
```

Aliases can be combined together. For example, before you clean up branches, you may want to switch to the target branch and update that branch from the remote server first.

```
[alias]
    bdone = "!f() { git checkout ${1-master} && git pull --
   rebase && git bclean ${1-master}; }; f"
```

Note how the `bdone` alias makes use of the `bclean` alias. You might use the `bdone` alias right after you push a branch that then gets merged by someone on GitHub.com. After the branch is merged, you can run git `bdone`, and the alias will switch you back to `master`, run a git pull, and then run the `bclean` alias.

TIP Adding aliases you find from elsewhere can be a pain, though. This post by one of the authors describes a quick way to include a bunch of aliases into your Git config file: `https://haacked.com/archive/2019/02/14/including-git-aliases/`.

Run Tests Automatically

Writing automated tests for code, such as unit tests, is an essential skill for professional software developers. It helps improve the design of code and provides a safety net when making changes to code.

When working with a code base with a lot of tests, it's not uncommon to forget to run tests often. And even if you do run tests often, it’s a distinct step: Make some changes and then run a command to run your tests.

A great way to improve your development workflow is to automate the test runs. Many tools will automatically run tests when your code changes. Here’s a short list of tools for various platforms:

- NCrunch for .NET
- Guard-test for ruby
- Tsc-watch for TypeScript
- Cargo-testify for Rust
- Wallaby.js for JavaScript

Many of these tools are smart about only running the tests affected by the changed code. That way, every change doesn’t end up running the entire test suite.

Take Breaks

One simple effective method for improving your workflow, albeit one that is neglected by many developers, is to take regular breaks. Writing code may feel like a sedentary task, but any activity done over an extended period of time can put a lot of stress on the body. The human head is pretty heavy (some heavier than others). Holding it upright can put a lot of stress on the back and neck. According to DataHand, a maker of ergonomic keyboards:

At the end of an average eight-hour workday, the fingers have walked 16 miles over the keys and have expended energy equal to the lifting of 1 1/4 tons.

Taking regular breaks to stretch your hands, arms, and back can help you remain healthy and thus more productive.

However, that’s not the only benefit of taking breaks. Many people are fans of the Pomodoro Technique developed by Francesco Cirillo in the late 1980s. This technique breaks work down into intervals, traditionally 25 minutes, with a short break of around 3 to 5 minutes in between. Each interval is known as a Pomodoro, Italian for tomato. Why a tomato? Apparently, Cirillo used a tomato-shape timer in college.

After four pomodoros, you take a longer break (15 to 30 minutes). The benefit of this technique is not only to enforce that you rest your body, but it has the added benefit of helping you maintain focus. The general idea is that during a pomodoro, you are intently focused on work. You generally close all other distractions. You can use some of the longer breaks to do things like check emails and browse the web, if you need to. But during the 25-minute pomodoro, you should be intently focused on work. Practitioners swear by the increased focus and flow the technique encourages. You can find many examples of pomodoro timer applications on the web.

Prototype User Interfaces

Whether it’s a web or desktop application, building a user interface can be very time-consuming. And it’s difficult to know how usable an interface will be until you put a human in front of it to try it out.

One tool that saves a lot of time when building an interface is a rapid prototyping tool, such as Balsamiq or Invision. This is by no means an exhaustive list of such tools.

The benefit of these tools is they make it possible to build mock-ups of a Graphical User Interface (GUI) in a short amount of time. Some tools even make it possible to add a bit of interactivity so that you can put the interface in front of a person and run some informal usability tests. You can get a lot of valuable feedback by simply asking people questions like “How would you accomplish a task on this screen? What would you click next?”

TIP

Making changes to the interface to respond to such feedback is much faster than if you had written a bunch of code. Once you’ve run through a few iterations with the mock-ups, you can build the actual GUI with more confidence that you’re on the right track.

Scaffold Apps with Yeoman

Starting a new application from scratch can be time-consuming. A lot of boilerplate code goes into setting up a real production application, and the boilerplate is different depending on the type of app and what the app does.

Yeoman is a tool for scaffolding modern web apps. To install it, run the following command:

```
$ npm install -g yo
```

This command adds the `yo` command to your machine. Yeoman works with generators, which are essentially plugins to the `yo` command, that add support for a given project type. A huge ecosystem of generators is out there.

For example, suppose that you want to build an extension for Visual Studio Code (VS Code). You would start by installing the generator for VS Code:

```
$ npm install -g yo generator-code
```

To run the generator, you run the `yo` command with the generator name:

```
$ yo code
```

The generator prompts you to answer some questions about the project to generate, such as specifying a project name. At the end, the generator takes your answers and scaffolds a project folder with a working VS Code extension.

Chrome Web Developer Tools

If you develop browser-based applications for the web or for the Desktop via Electron, no tool is probably more useful than the Chrome Web Developer Tools. To launch these tools, use the shortcut ⌘ + Option + I on the Mac or Ctrl + Shift + I on Windows.

You can also launch the developer tools by right-clicking on an element of any web page and choosing the Inspect menu option. When the developer tools open, select that element in the Elements tab.

The Elements tab of the developer tools allows you to explore and manipulate the DOM. You can also manipulate the CSS. This provides a nice way to debug CSS problems because it gives you instant feedback on your CSS changes.

The Console tab lets you run JavaScript commands in the context of the current web page.

The Sources tab lists all the scripts that are loaded in the context of the page. This list can be eye-opening when you go to a website you use often and look at this tab. A given web page can have a large number of scripts running.

The Network tab shows all the network requests that were made to render the page, the size of the requests, how long they took, and when they happened in relation to each other. This information is useful for debugging issues where a large request is causing delays in rendering a page.

The Performance and Memory tabs are useful for profiling execution time and memory usage of a page.

StackOverflow

StackOverflow.com is a question and answer website for developers. Since its creation in 2008, it's had a huge impact on the developer community. A big part of its popularity is due to the gamification techniques it employs to maintain high-quality questions and answers.

For example, questions and answers can be up voted and down voted. Answers that receive the most up votes are displayed directly under the question so that people who find the question later don’t have to wade through a ton of answers to find the best answer. If the poster of a question accepts an answer, that answer floats to the top (unless the poster also answered the question), regardless of the number of upvotes. This setup sometimes causes a situation where a better answer with more up votes is displayed before the accepted answer.

Asking questions, answering them, and having questions or answers voted up all contribute to your reputation points. As your reputation points increase, you gain more privileges on StackOverflow, such as being able to edit questions and answers to collectively improve them like a wiki.

If you’re stuck on a programming task, StackOverflow is often a good place to start searching for an answer to your question.

Code Analysis Tools

A wide range of tools can analyze code for potential problems and potential improvements. When used properly, these tools can save you a lot of time and headache.

Linters are a class of tools named after a Unix utility named Lint that analyzes C code to flag bugs, style errors, and potential problems without having to run the code. While Lint is the original tool, many linting tools exist for different programming languages, such as JSLint for JavaScript and ruby-lint for Ruby.

Static analysis tools are similar to linters but work against statically typed languages. These tools take advantage of type information in the source code to find issues in the code that aren't syntax errors (which the compiler would already catch) but may cause problems down the road. For example, static analysis tools can flag code that may exhibit poor performance in certain situations. Examples of static analysis tools include FxCop for .NET and Coverity for Java.

Some tools surface metrics about your code that may be correlated to quality, such as measuring cyclometric complexity. Cyclometric complexity refers to the number of different execution paths through a piece of code. A method with a very large cyclometric complexity can be hard to understand and prone to bugs.

Some tools, such as Code Climate Quality, are available in the GitHub Marketplace and can perform automated code reviews looking for common problems in code. This tool can identify files and sections of code that are changed frequently. Frequent changes often indicates that the code may have quality issues. Understanding where your code churns often helps you focus your attention on changes to that code.

Project Boards

Project boards are a useful way to visualize the progress and tasks for a project. Chapters 3 and 4 walk through setting up a project board along with project automation for a repository. For a given repository, a project board serves as a common source of truth. Everyone can look at a project board and have a good idea of the overall progress at a glance.

However, you may consider using individual project boards that are not associated with a single repository as a means of managing your overall set of tasks on GitHub. You can go to `https://github.com/new/project` to create a new personal project board.

Project boards come with a limited set of automations. To really customize your workflow, you may want to create a GitHub Action (see Chapter 14). With GitHub Actions and the GitHub API, it's possible to create project board automations for nearly any workflow you can think of.

GitHub project boards are not the only option for a Kanban-style board that works with GitHub. Trello (see Chapter 14) is another option. A few other options with deeper integration with GitHub include ZenHub, waffle.io, and HuBoard.

Regardless of the one you pick, a good project board is a helpful addition to any software developer's workflow.

Chapter 20

Ten Tips for Being an Effective Community Member

IN THIS CHAPTER

- **Discovering how to be a participant that OSS maintainers value**
- **Fostering a community that brings the best out of contributors**
- **Encouraging an overall community of respect and collaboration that is welcoming**

As of November 2018, more than 31 million people are on GitHub. That's a lot of people. It can be difficult to stand out among such a large sea of people. But the truth is, most people starting out on GitHub don't really understand how to be effective. They don't understand the rules of the road to being a great member of the community.

In this chapter, we compile ten tips that will help you be a wonderful community member — the type of person that every maintainer is excited to have involved with their repositories.

Be Respectful and Kind

Maintaining a repository can be a frustrating affair — especially when it's a popular repository and you're volunteering your free time to an open source project and you're about to answer the same question for the hundredth time. It's understandable that you may be very curt to the next person who asks the same question. It's a waste of your time, and they didn't do their due diligence to search to see whether the question was already answered.

WARNING Resist the temptation to lash out. For that person, it may be the first time they've ever created an issue. They may not have read this chapter yet and learned how to be effective on GitHub. Your response will set the tone for their experience of your project and, perhaps, of GitHub as a whole.

Be respectful and kind, and you may win over someone who will become a lifelong contributor. And even in the face of rudeness, remember that we all have bad days. You've probably done the same in the past yourself. The Internet can be very impersonal when communicating in writing. Sometimes a little kindness at the right moment reminds people that a human is on the other side and there's no need to be rude.

REMEMBER But to be clear, killing them with kindness is fine for the occasional rude behavior, but you should not have to tolerate abusive behavior.

Report Bad Behavior

Whether bad behavior is directed at you or others, it's important to the health of a community that we remain vigilant and report it. If you encounter or witness abuse on GitHub, report it at `https://github.com/contact/report-abuse`.

TIP It can be difficult to know the difference between someone just being slightly rude and someone being abusive. GitHub's Terms of Service help spell it out. If you're unsure, know that GitHub's support people are well trained to draw this distinction and will not react in a knee-jerk fashion to your report.

Abuse isn't the only type of issue you may want to report to GitHub. You can report harmful content, privacy concerns, and copyright claims from GitHub's contact page at `https://github.com/contact`.

Write Good Bug Reports

For a repository maintainer, few things are more frustrating than an issue that looks something like this:

The thagomizer isn't working, fix it! I tried it out, and it doesn't work.

This example may seem extreme, but vague bug reports that are not actionable are unfortunately quite common. It doesn't take too much effort to write a bug report that is helpful.

TIP When you come across a bug report where you have some expertise, don't respond with "Works on my machine" with no other follow-up. Confirming that it works on your machine is actually beneficial, but it is more helpful to offer suggestions for what may be different between your machine and the person who filed the bug report. For example, you may want to let the person know which version of the software you have installed or if you have any other setup that may affect the situation.

Sometimes, the best bug report is the one that isn't written. Before you create an issue, search the issue tracker to see whether someone else already reported the bug. If they did, you may want to add further details to the issue if you see anything missing.

Assuming you didn't find the issue, it's time to create a new issue. If the repository has an issue template, you should follow the template as closely as possible.

If is the repository has no template, the following is a good format to follow:

1. **Describe the observed behavior.**

   Try to be objective and communicate facts, not opinions.

2. **Describe what you expected to happen.**

   How did the observed behavior differ from your expectations?

3. **Describe detailed repro steps.**

   This step is the most important part of the bug report. Describe step by step how someone else can reproduce the problem. Try to make the repro steps as minimal as possible. When you encountered the bug, you may have taken ten steps to get there. But it may be possible to reproduce the bug in only five steps. Spending a little time to make sure that you remove any extraneous steps goes a long way into making a good bug report.

4. **Describe the repro environment.**

   This step is where you describe the environment where you reproduced the bug such as the operating system and browser. While it's not necessary to try to reproduce the bug in other environments, repository owners are very appreciative if you do and report on the results.

Be Responsive

No matter how good your issue write-up is or your pull request code is, chances are the repository maintainers will have some follow-up questions. It's particularly frustrating for a maintainer to ask for more information only to be greeted by crickets. If you submit an issue, don't ghost on it. Make sure you make time to follow up and respond to questions from the maintainers.

REMEMBER And the shoe fits both ways. If you are a maintainer, try and be responsive to people who submit issues and pull requests. In some cases, using an automated response to an new issue or pull-request is appropriate if your repository is particularly busy. For example, you could use a Probot app to automatically respond to new issues and pull requests with a note letting them know you plan to look at it but that it may take some time.

Being responsive doesn't necessarily mean you take care of everything right away. It means that you set expectations right away. On either side of the comment, whether you're a contributor or a maintainer, the only way for the other person to know when to expect changes is if you tell them. Working on remote, asynchronous projects depends heavily on communication. When you're in an office you can see whether someone doesn't come in to work for a while and know they are probably on vacation; on GitHub, you don't have that contextual awareness.

TIP As an added form of communication, keep your profile status up to date if you do plan on going on vacation. For more detail, see Chapter 16.

Submit Pull Requests to Correct Documentation

The campsite rule states that one should leave a campsite in better condition than they found it. It's a good rule to follow not only with campsites, but also with documentation.

Good documentation is often the weak point with an open source projects. Many OSS projects have very few volunteers, and much of their time is taken working on the actual code. This lack of support makes repository maintainers especially appreciative when someone comes along and contributes to the documentation.

Getting involved with improving a repository's documentation is also a great way to dip your toe into OSS. If you happen to find an error or something missing in a project's documentation, consider submitting a pull request to the project fixing the error. This contribution leaves the overall OSS ecosystem better off than it was before.

REMEMBER One of the most challenging pieces of documentation to keep up to date is the Getting Started docs that contributors are meant to follow to first get the project setup on their machine. It’s challenging because the maintainers rarely set up the project brand new, unless they get a new machine. As you’re following the steps, don’t be afraid to open a pull request on this documentation if you had to do something different. It can be especially helpful if instructions are different on Mac versus Windows versus Linux and you’re on a machine that the project doesn’t have documentation for yet.

Document Your Own Code

Documenting your own code goes a long ways toward making it more accessible to others. Trying to use unfamiliar code can be challenging and time consuming. Good documentation can save people a lot of time and get them up and running with your code quickly. If only 12 people use your code and you save them each one two hours of hassle, that’s a full day saved!

Depending on the platform, many tools generate documentation from comments in code. Javadoc is a famous example for Java code. It requires that you comment public methods and classes with a standard format. By following the format, you can use the Javadoc tool to generate HTML files. JSDoc is another one for JavaScript.

In addition to code documentation, consider other documentation such as the ones we cover in Chapter 9. For example, every repository should have a `README.md` file that describes what the repository does. It should also have a `CONTRIBUTING.md` file that describes how to contribute.

Give Credit Where It's Due

WARNING Most open source licenses require proper attribution if you make use of the code in your own project. If you use some source code from an open source project, giving credit where it's due is a legal matter and required by the license.

But to be an effective community member, credit doesn't end with attribution to comply with a license. In many situations, giving credit demonstrates that you are a classy person.

For example, if someone contributes a feature or bug fix, mentioning the person who fixed it in your release notes is a good idea. For example, GitHub Desktop makes a point to thank people using its GitHub handles and links to the pull request that contributed a fix in its release notes at `https://desktop.github.com/release-notes/`.

Another great way to give credit is to mention the person who opened an issue that you may have fixed with a pull request. The pull request description will most likely link to the issue, but calling out the person who found the issue in the description is a great way to encourage others to continue to help find bugs.

Help Get the Word Out

Many open source projects are small and relatively unknown. If you find a project useful, help get the word out. It not only benefits the project, who may get an influx of new users and contributors, but it benefits the people you tell who may need that very tool.

REMEMBER You don't need a huge platform to help people get the word out. Maybe your Twitter follower count is relatively small. Don't let that stop you. The power of network effects can sometimes help a message really take off.

If you write blog posts or publish YouTube videos, you can also mention different projects, your impressions, and maybe even a tutorial on them on these mediums. The goal here is to help people and projects meet.

Be Proactive and Mentor Others

GitHub’s community is growing rapidly. New developers are being minted every single day. What this means is a lot of beginners will be on the site. Over time, you will start to accumulate experience that would be very valuable to one of these beginners. Be proactive and offer to mentor others.

For example, if you maintain an open source repository and see that someone is struggling to make a contribution, offer to walk them through the process. Point them to resources and help them along. If you help two people become proficient contributors and community members and they each help two people down the road and so on, you could end up having a huge impact on the community.

Sometimes it can even be effective to jump on a video chat and help someone debug their code in real time! If you’re comfortable with this approach, it can be a great way to meet new people and see for yourself how someone new approaches your project. It may provide insight into how to improve your documentation. One of our favorite video chat services to use for this purpose is `https://appear.in`. It’s a free, in-browser video chat service that allows for multiple attendees and allows you to share your screen.

Contribute Outside of GitHub

Many open source projects have a lot of activity outside of GitHub. Contributing to the community can go beyond creating issues and opening pull requests. For example, as you gain expertise in a topic, consider heading over to StackOverflow.com and answering questions on that topic. Many open source projects have chat rooms associated with the project in Slack or Gitter. It’s a benefit to others if you head over there and offer your ideas.

Talk to maintainers about other ways you can support their work. As you grow in your career as a software developer, you will pick up skills that are valuable to an OSS project. For example, you may help them figure out how to sign a package using Let’s Encrypt. You may help them register a domain name and pay for it. You may help them navigate setting up a Docker container so that others can try out their project with less setup fuss.

Whatever it is, don’t be shy in offering your skills in support of open source projects, especially those that you benefit from.

Index

A

- `-A` flag, 119
- Abdalla, Safia (Microsoft developer), 307
- accessing
  - existing GitHub projects, 77–83
  - GitHub.com repository, 78–79
- Account menu (GitHub.com), 30
- Account settings (GitHub.com), 19
- Action buttons (Code tab), 47
- Actions (GitHub)
  - about, 260
  - creating workflows, 260–262
  - GitHub Marketplace and, 267
  - testing, 262
- adding
  - code of conduct, 173
  - contributor guidelines, 173
  - licenses, 170–172
  - reviewers, 138
  - sections to personal websites, 91–92
  - sections to websites, 69, 91–92
- after-hour conference events, 290

alerts, of repositories, 205

animated gif, 143

Anita Borg institute (website), 294

appear.in (website), 323

Apple source control integrations, 247

applications (apps)

- AppVeyor, 271
- code quality, 271–272
- Codecov, 271–272
- common, 270–274
- connecting with GitHub.com, 24
- Crowdin, 272
- customizing, 257–259
- Dependabot, 273
- dependency management, 273
- GitHub, 255–259
- Greenkeeper, 272–273
- GUI, 122–128
- hosting, 256
- Installed GitHub, 24, 198
- installing, 259
- Khan Academy Kids (website), 306
- learning, 274
- listing on GitHub Marketplace, 268–270
- localization, 272
- monitoring, 272–273
- MyGet, 273

  - OAuth, 25
  - Probot Glitch, 256–257
  - scaffolding, 313–314
  - Terminal, 9–14, 137
  - TestQuality, 274
- applying labels, 82–83
- AppVeyor app, 271
- archiving projects, 185–186
- Asset Store (Unity), 239, 242
- assignees, specifying, 139
- assigning CODEOWNERS, 201–204
- @ symbol, 130–132
- Atom
  - about, 35–38
  - components of, 74–75
  - documentation, 38
  - following GitHub package for, 234
  - installing, 36
  - opening repo in, 74
  - using GitHub for, 229–235
  - website, 36, 230
- Atom Flight Manual (website), 38, 74, 234
- Atom GitHub repo (website), 84
- audience, knowing your, 140
- Audit log setting, 198
- authorized GitHub apps, 24
- authorized OAuth apps, 24

automatic testing, 311–312

automation settings, changing, 58

B

banning people, 175–176

Beams, Chris (blogger), 132

behavior, reporting bad, 318

billing

- for apps, 268
- in GitHub Marketplace, 266–267
- setting for, 21–22, 198

bio, for profile, 277–278

Blocked users setting (GitHub.com), 23

blocking users, 176–177

blogs

- about, 306–307
- creating, 92–93
- writing posts, 70

branch Compare page, opening, 137

branch list

- Atom, 75
- GitHub Desktop, 72

branches

- about, 17
- confirming publication of, 86–89
- creating, 83–86, 134

- creating for contributions, 83–86
- branching
  - by collaborator, 14–15
  - by experimentation, 16
  - by feature, 15–16
- brand and listing, for apps, 267
- breaks, taking, 312–313
- Brooklyn JS (website), 286
- browser, opening to repository, 124
- browser extensions
  - about, 251–252
  - GitHub Selfie, 254
  - refining GitHub, 252–253
  - website, 252
- `bug` label, 82, 179
- bug reports, writing, 318–319
- Build (website), 288
- “Build Your Own Octocat” app, 29
- “Building an Inclusive Code Review Culture” (website), 151

C

- call to action, including in pull requests, 143–144
- Cargo-testify for Rust, 312
- changes
  - staging, 117–118
  - suggesting to pull requests, 148–150

changes list (GitHub Desktop), 73–74

changing

- automation settings, 58
- `README.md`, 48–53
- taglines for personal websites, 91
- taglines for websites, 69, 91
- titles of personal websites, 91
- titles of websites, 69, 91
- username, 19

chat

- about, 212
- as a category for tools and apps, 265
- installing GitHub app for Slack, 212–214
- subscribing to repositories in Slack channel, 214–216
- trying GitHub Slack integration, 217–219

Cheat Sheet (website), 3

checking out pull requests, 230–233

child teams, creating, 199–200

Chrome Web Developer Tools, 314–315

CI (continuous integration), 145, 271

clarity

- of purpose for pull requests, 141
- reviewing code for, 146

`clear` command, 13

cloning

- about, 16, 98
- repo in GitHub Desktop, 71–72

- repositories, 99–100
- without forking, 107–110

closing
- issues, 60–62, 83
- stale issues, 83

coauthors, giving credit to in commit messages, 130–132

code
- about, 189
- analysis tools for, 315–316
- assigning CODEOWNERS, 201–204
- best practices for inner-sourcing, 204–208
- Code tab, 47
- commenting on, 146–148
- committing, 113–132
- committing from editor, 132
- committing with GitHub Desktop, 122–128
- creating GitHub organizations, 190–191
- creating parent/child teams, 199–200
- creating teams within GitHub organizations, 195–196
- documenting, 321
- inviting members to GitHub organizations, 191–192
- managing members of GitHub organizations, 193–195
- milestones for larger projects, 207–208
- privacy of, 189
- pushing to GitHub, 134–135
- repository insights, 204–207
- setting GitHub organization settings, 197–199

  - team discussions, 200–201
  - teams, 199–204
  - using GitHub organizations, 190–199
  - using project boards within GitHub organizations, 196–197
  - viewing repositories for GitHub organizations, 192–193
  - writing, 113–132
- Code Climate Quality, 316
- code editor (Atom), 74
- code frequency, of repositories, 205
- code of conduct
  - about, 290
  - adding, 173
  - enforcing, 175–177
  - reading, 165
  - website, 307
- code quality apps, 271–272
- code quality category, for tools and apps, 265
- code review category, for tools and apps, 265
- "Code Review Like You Mean It" (website), 151
- Code tab, 45, 46–47
- Codecov app, 271–272
- `CODE_OF_CONDUCT.md` file, 80, 165, 173, 178
- CODEOWNERS, assigning, 201–204
- collaboration
  - about, 211
  - chat, 212–219
  - integrating GitHub with Trello, 219–225

  - managing notifications with Octobox, 225–227
- collaborative coding environment, setting up, 27–38
- collaborator, branching by, 14–15
- collections, 160
- command line, 10
- command line tool, installing, 123–124
- Command Palette command, 233
- commands
  - `clear`, 13
  - `commit`, 12, 13
  - `explorer <path>`, 11
  - `git bisect`, 117
  - `git commit`, 118
  - `git push`, 135
  - `git status`, 118, 119
  - `/github subscribe`, 219
  - `/github subscribe owner/repo[feature]`, 219
  - `log`, 13
  - `open <path>`, 11
  - `pwd`, 10
  - `status`, 11
  - `yo`, 314
- comments
  - on code, 146–148
  - responding to, 83
- `commit` command, 12, 13
- commit messages

- using conventions in, 129–132
- writing, 120–122

commits
- about, 17
- creating, 116–120
- defined, 113
- of repositories, 205

committing
- code, 113–132
- code from editor, 132
- code with GitHub Desktop, 122–128
- files, 118–120
- in GitHub Desktop, 125–128
- multiple files, 119–120

community, of repositories, 205

community members, being effective, 317–323

community places, external, 304

Compare page, opening, 137

Compare & pull request button, 104, 135

conference session tracks, 289

conferences, major, 288

Confirm Merge dialog box, 55

confirming branches are published, 86–89

Console tab, 314

Contact page, 299, 318

continuous integration (CI), 145, 271

continuous integration (CI) category, for tools and apps, 266 contributing

    changes to upstream, 104–107

    finding places for, 161–163

    outside GitHub, 323

    setting contributor expectations, 166–167

    surveying projects for, 164–165

contributing code guide, reading, 164–165

`CONTRIBUTING.md` file, 80, 164, 178

contribution activity, on profile, 281

contribution graph

    defined, 32

    on profile, 279–281

contributions

    creating branches for, 83–86

    preparing, 83–90

contributors

    adding guidelines for, 173

    of repositories, 205

conventions

    reviewing code for, 146

    using in commit messages, 129–132

Conversation tab, for pull requests, 145

correctness, reviewing code for, 146

Coverity for Java, 316

Create a new project dialog box, 56

Create a New Repository dialog page, 42

creating
    blogs, 92–93
    branches, 83–86, 134
    branches for contributions, 83–86
    commit, 116–120
    Git Alias, 111–112
    GitHub Action workflows, 260–262
    GitHub organizations, 190–191
    issues, 56–60, 69–71
    issues for websites, 69–71
    open source repositories, 169–173
    parent/child teams, 199–200
    personal websites, 91–93
    Probot Glitch apps, 256–257
    project boards, 56–60
    pull requests, 139–140, 230–233, 244–246
    pull requests in Visual Studio, 244–246
    repositories, 113–114
    teams within GitHub organizations, 195–196
    websites with GitHub Pages, 77–93
credit, giving, 130–132, 321–322
Crowdin app, 272
customizing apps, 257–259
cyclometric complexity, 316

D

Dependabot app, 273

dependency graph, of repositories, 205

dependency management app, 273

dependency management category, for tools and apps, 266

dependency security vulnerabilities, 205

deployment category, for tools and apps, 266

description

- in commit messages, 121
- of pull requests, 142

Description and topics (Code tab), 47

Desktop (GitHub)

- about, 33–34
- cloning repo in, 71–72
- committing code with, 122–128
- committing in, 125–128
- components of, 72–74
- installing, 33–34
- open source project (website), 80
- publishing repositories in, 124–125
- setting up, 34–35
- tracking repositories in, 123–124

`<details>` tag, 142

Developer settings, 25, 198

diff view, 127

Discover page, 157

documentation

- for code, 321
- project-specific, 302–303

- writing, 178–179

drafting pull requests, 309–310

`duplicate` label, 179

duplicating, 98

E

editor, committing code from, 132

Electron, 35

Elements tab, 314

Email setting (GitHub.com), 19–20

emojis

- in commit messages, 122
- using in commit messages, 129

ending projects, 185–187

`enhancement` label, 180

Enterprise tier, 190–191

events

- about, 285
- finding funding for, 293–294
- Git Merge, 292
- GitHub Constellation, 292
- GitHub Satellite, 291
- GitHub Universe, 291
- hackathons, 287–288
- major conferences, 288
- meet-ups, 286

- regional conferences, 286–287
- speaking at, 292–293
- types of, 286–288
- user groups, 286
- what to expect at, 288–290

Events section (Explore page), 160

experimentation, branching by, 16

Explore link (GitHub.com), 30

Explore page

- about, 156
- collections, 160
- Events section, 160
- headline section, 156
- Marketplace apps, 160
- Popular Topics section, 158–160
- repositories, 157
- starring repositories, 161
- trending repositories, 157–158

`explorer <path>` command, 11

extensible, 35

extensions

- GitHub for Unity, 242
- GitHub for Visual Studio, 246
- GitHub for VS Code pull requests, 238

extensions, browser

- about, 251–252
- GitHub Selfie, 254

- refining GitHub, 252–253
- website, 252

external community places, 304

F

feature, branching by, 15–16

feature/enhancement label, 82

file list (Atom), 74

files

- `CODE_OF_CONDUCT.md`, 80, 165, 173, 178
- committing, 118
- committing multiple, 119–120
- `CONTRIBUTING.md`, 80, 164, 178
- `README.md`, 48–53, 80, 178
- reviewing changed, 146

finding

- funding for events, 293–294
- places to contribute, 161–163
- resources for GitHub Pages, 76
- Terminal app, 9

focus, of pull requests, 141

following users, 282–283

fork, 16–17, 44, 206

Fork button, 45, 101

forking

- about, 97–98

- cloning without, 107–110
- GitHub repositories, 97–112
- repositories, 100–112

Frazelle, Jesse (blogger), 260

funding, finding for events, 293–294

FxCop for .NET, 316

G

gif, animated, 143

Git

- on GitHub, 16–17
- as a type of version control system, 8–16

Git Alias

- about, 109, 111–112, 311
- opening branch Compare page, 137
- for opening GitHub.com from Terminal, 137

`git bisect` command, 117

`git commit` command, 118

Git Docs (website), 14

Git documentation (website), 132

Git for Windows (website), 9

Git index, 118

Git Large File Storage (Git LFS), 22

Git Merge, 292

Git panel toggle button (Atom), 75

`git push` command, 135

`git status` command, 118, 119

Git tab (Atom), 37

GitHub. *see also specific topics*

- about, 7
- apps, 255–259
- blog, 4
- contributing outside, 323
- Git on, 16–17
- integrating with Trello, 219–225
- pushing code to, 134–135
- refining, 252–253
- using for Atom, 229–235
- using for IntelliJ, 248–250
- using for Unity, 239–242
- using for Unity in Unity, 240–242
- using for Visual Studio, 243–246
- using for Visual Studio Code, 235–238
- using for XCode, 246–248
- website, 4

GitHub Actions

- about, 260
- creating workflows, 260–262
- GitHub Marketplace and, 267
- testing, 262

GitHub API (website), 25

GitHub apps, 25, 212–214

GitHub blog (website), 307

GitHub Community Forum, 25, 307–308

GitHub Constellation, 292

GitHub contact page (website), 25

GitHub Desktop

  about, 33–34

  cloning repo in, 71–72

  committing code with, 122–128

  committing in, 125–128

  components of, 72–74

  installing, 33–34

  open source project (website), 80

  publishing repositories in, 124–125

  setting up, 34–35

  tracking repositories in, 123–124

GitHub Desktop User Guides (website), 72

GitHub Developers Program (website), 197

GitHub Education community forum (website), 308

GitHub for Unity extension, 242

GitHub for Visual Studio extension, 246

GitHub Git Cheat Sheet (website), 14

GitHub Guides (website), 299

GitHub Help (website), 23, 25, 93, 298

GitHub Learning Labs, 3, 25, 274, 300–301

GitHub Marketplace

  about, 265–266

  billing, 266–267

  code quality apps, 271

- common apps to install, 270–274
- continuous integration (CI) apps, 271
- dependency management apps, 273
- on GitHub.com, 30
- learning, 274
- listing apps on, 268–270
- localization apps, 272
- monitoring apps, 272–273
- testing software, 273–274
- vetting process, 267–268

GitHub organizations
- about, 190
- creating, 190–191
- creating teams within, 195–196
- inviting members to, 191–192
- managing members of, 193–195
- settings for, 197–199
- using project boards within, 196–197
- viewing repositories for, 192–193

GitHub package, following for Atom, 234

GitHub Pages
- about, 64
- creating websites with, 77–93
- finding resources for, 76

GitHub panel toggle button (Atom), 75

GitHub repositories
- about, 16

cloning, 71–72, 99–100
cloning in GitHub Desktop, 71–72
Code tab, 46–47
components of, 44–47
creating, 113–114
creating issues for websites, 69–71
Explore page, 157
exploring, 44–47
finding resources for GitHub Pages, 76
forking, 97–112, 100–112
GitHub, 35
insights of, 204–207
issues, 56–62
linking, 70, 93
making public, 173–174
merging pull requests, 53–56
modifying README.md, 48–53
open source, 38, 169–173
opening browser to, 124
opening in Atom, 74
pinned, 161, 278–279
project boards, 56–62
publishing in GitHub Desktop, 124–125
setting up, 41–44
setting up a personal website, 66–69
setting up local environment, 71–75
setup, 63–76

- starring, 161, 281–282
- subscribing to in Slack channel, 214–216
- tabs, 45–46
- top information, 44–45
- tracking in GitHub Desktop, 123–124
- trending, 157–158
- turning a project repo into a website, 64–66
- turning into a website, 64–66
- verifying permissions for, 79–80
- viewing for GitHub organizations, 192–193
- website, 64, 234

GitHub Satellite, 291

GitHub search, 282, 283

GitHub Selfie, 254

`/github subscribe` command, 219

`/github subscribe owner/repo[feature]` command, 219

GitHub tab (Atom), 37

GitHub Universe, 291

GitHub.com

- exploring, 27–31
- opening from Terminal, 137
- personalizing your account, 18–25
- signing up for, 17–18

GitHub.com repository, accessing, 78–79

git-school (website), 14

Glitch, 256

GNU General Public License v3.0 License, 43 `good first issue` label, 82, 180

Google Docs, 7

GoRuCo (website), 287

GPG key, 22–23

Grace Hopper Celebration, 294

Graphical User Interface (GUI), 313

Greenkeeper app, 272–273

Guard-test for Ruby, 312

GUI (Graphical User Interface), 313

GUI application, 122–128

H

hackathons, 287–288

hallway tracks, 290

headline section (Explore page), 156

help documentation, 298–299

`help wanted` label, 82, 180

history list (GitHub Desktop), 74

home page, 28

Hopper Program (website), 294

hosting apps, 256

“How to write the perfect pull request” (website), 143

HTML, 253

Hubbers, 278

HuBoard, 316

I

icons, explained, 3

IDE (integrated development environments), 239

idioms, reviewing code for, 146

images, for pull requests, 142–143

improving development workflow, 309–316

including call to action in pull requests, 143–144

inner-sourcing

- best practices for, 204–208
- defined, 189

in-person training, 301–302

insights, of repositories, 204–207

Insights tab, 46

Installed GitHub apps, 24, 198

installing

- apps, 259
- Atom, 36
- command line tool, 123–124
- GitHub app for Slack, 212–214
- GitHub Desktop, 33–34
- GitHub power-up, 220–222

integrated development environments (IDE), 239

integrating Trello and GitHub, 219–225

IntelliJ, using GitHub for, 248–250

interacting with pull requests in VS code, 237–238

Internet resources

- Anita Borg institute, 294
- appear.in, 323

Apple source control integrations, 247

Atom, 36, 230

Atom documentation, 38

Atom Flight Manual, 38, 74, 234

Atom GitHub repo, 84

Atom packages, 37

Atom project forum, 304

Beams, Chris (blogger), 132

blocking users, 177

branching, 15

Brooklyn JS, 286

browser extensions, 252

Build, 288

“Build Your Own Octocat” app, 29

“Building an Inclusive Code Review Culture,” 151

Cheat Sheet, 3

VS Code, 303

Code of Conduct, 307

“Code Review Like You Mean It,” 151

CODEOWNERS, 204

Contact page, 299

dependency security vulnerabilities, 205

descriptions, 160

Discover page, 157

documentation example, 179

example commit message, 121

example commits, 122

Explore page, 156

Git Alias, 111

Git Docs, 14

Git documentation, 132

Git for Windows, 9

Git Merge, 292

GitHub, 4

GitHub API, 25

GitHub blog, 4, 307

GitHub Community Forum, 25, 307

GitHub Constellation, 292

GitHub contact page, 25

GitHub Contact page, 318

GitHub Desktop, 35

GitHub Desktop open source project, 80

GitHub Desktop User Guides, 72

GitHub Developers Program, 197

GitHub Education community forum, 308

GitHub for Unity extension, 242

GitHub for Visual Studio extension, 246

GitHub for VS Code pull requests extension, 238

GitHub Git Cheat Sheet, 14

GitHub Guides, 299

GitHub Help, 23, 25, 93, 298

GitHub Learning Labs, 3, 25, 300

GitHub Marketplace landing page, 268

GitHub repo, 64, 234

GitHub repository, 35

GitHub Satellite, 291

GitHub search, 282, 283

GitHub Selfie, 254

GitHub Universe, 291

git-school, 14

Glitch, 256

GoRuCo, 287

Hopper Program, 294

“How to write the perfect pull request,” 143

Jekyll, 64

JSConfHawaii, 287

Khan Academy Kids app, 306

learngitbranching, 14

“The Life of a GitHUb Action,” 260

list of apps, 258

logos, 160

Markdown GitHub Guide (website), 64

Marketplace apps, 267

Meetups.com, 286

Microsoft Imagine Cup, 287

Microsoft Open Source Code of Conduct, 165

Microsoft organization home page, 192–193

.NET Fringe, 287

.NET São Paulo, 286

Node.js repository, 160

Octobox, 225

Octocat history, 29

Octodex of Octocats, 29

Open Source Definition (OSD), 169

open source repository, 38

Oscon, 288

outlearning labs, 76

Popular Topics section (Explore page), 158

public Slack, 304

`README.md` template, 48

release notes, 322

reporting abuse on GitHub, 318

reviewing pull requests, 81

Ruby tutorial, 304–305

for saved replies, 183

SD Ruby, 286

search help, 30

setting up PowerShell, 9

setting up SSD, 23

Slack, 212

software licenses, 43

starred repositories, 161

team discussion home page, 200

training, 302

Trello, 220

trending repositories, 157

Try.NET, 305

Up for Grabs, 162

  - version control, 8
  - Visual Git Cheat Sheet, 14
  - Visual Studio Code, 235
  - Visual Studio Code contributing guidelines, 164
  - Visual Studio Code repository, 164
  - watching repositories, 45
  - WWDC, 288
  - XKCD commit, 117
- `invalid` label, 180
- inviting members to GitHub organizations, 191–192
- issue references, using in commit messages, 129–130
- issues
  - closing, 60–62
  - creating, 56–60, 69–71
  - creating for websites, 69–71
  - in GitHub.com, 30
  - labeling, 179–180
  - managing, 179–184
  - reading through, 81
  - repository, 56–62
  - stale, 83
  - templates for, 181–183
  - triaging, 180–181
  - viewing, 233–234
- Issues tab, 45

J

Jekyll, 64

Joomla, 98

JSConfHawaii (website), 287

JSDoc, 321

K

keyboard bindings, 37

keyboard shortcuts

    for hashes, 13

    opening browser to repository, 124

keynotes, 289

Khan Academy Kids app (website), 306

knowing your audience, 140

L

labels

    applying, 82–83

    for issues, 179–180

    specifying, 139

landmarks, 253

learngitbranching (website), 14

learning apps, 274

learning category, for tools and apps, 266

Learning Labs, 76

leveling up, 297–308

licenses, adding, 170–172

"The Life of a GitHub Action," 260

linking project repos, 70, 93

linters, 315

Linux, using command line in, 10

listing apps on GitHub Marketplace, 268–270

local environment, setting up, 71–75

localization apps, 272

localization category, for tools and apps, 266

`log` command, 13

logo (GitHub.com), 29

M

`-m` flag, 119

Mac

    installing command line tool, 123–124

    installing GitHub Desktop, 33

    using command line in, 10

Mambo project, 98

managing

    issues, 179–184

    members of GitHub organizations, 193–195

    notifications with Octobox, 225–227

Markdown GitHub Guide (website), 64

Markdown language, 42, 115

Marketplace (GitHub)

    about, 265–266

- billing, 266–267
- code quality apps, 271
- common apps to install, 270–274
- continuous integration (CI) apps, 271
- dependency management apps, 273
- on GitHub.com, 30
- learning, 274
- listing apps on, 268–270
- localization apps, 272
- monitoring apps, 272–273
- testing software, 273–274
- vetting process, 267–268

Marketplace apps (Explore page), 160

McEfee, Cameron (logo designer), 29

meet-ups, 286

Meetups.com, 286

Member privileges setting, 198

members

- inviting to GitHub organizations, 191–192
- managing of GitHub organizations, 193–195

mentoring, 322–323

merge conflicts, resolving, 89–90

merging pull requests, 53–56

Metadata bar (Code tab), 47

Microsoft Imagine Cup (website), 287

Microsoft Open Source Code of Conduct (website), 165

milestones

  - for projects, 207–208
  - specifying, 139
- MIT License, 43
- mobile category, for tools and apps, 266
- Moderation settings, 198
- modifying
  - automation settings, 58
  - `README.md`, 48–53
  - taglines for websites, 69
  - taglines on personal websites, 91
  - titles of websites, 69
  - titles on personal websites, 91
  - username, 19
- Mona Lisa Octocat, 29
- monitoring apps, 272–273
- monitoring category, for tools and apps, 266
- MyGet app, 273

N

- Nadella, Satya (CEO), 287
- NCrunch for .NET, 312
- needs investigation label, 82
- .NET Fringe (website), 287
- .NET São Paulo (website), 286
- network, of repositories, 206
- Network tab, 314
- Node.js repository, 160

notifications, managing with Octobox, 225–227

Notifications icon (GitHub.com), 30

Notifications setting (GitHub.com), 21

O

OAuth apps, 25

Octobox

- managing notifications with, 225–227
- website, 225

octocat, 29

Octodex of Octocats, 29

online coding tutorials, 304–305

online courses/tutorials, 305–306

Open Issue button, 233

`open <path>` command, 11

Open Source Community, 76

Open Source Definition (OSD), 169

Open Source Initiative (OSI), 169

open source repositories

- creating, 169–173
- website, 38

open source software (OSS)

- about, 17, 155, 169
- adding code of conduct, 173
- adding contributor guidelines, 173
- adding licenses, 170–172

  - archiving projects, 185–186
  - banning people, 175–176
  - blocking users, 176–177
  - creating open source repository, 169–173
  - ending projects, 185–187
  - enforcing code of conduct, 175–177
  - Explore page, 156–161
  - finding places to contribute, 161–163
  - issue templates, 181–183
  - labeling issues, 179–180
  - making repositories public, 173–174
  - managing issues, 179–184
  - responding with kindness, 175
  - saved replies, 183–184
  - setting contributor expectations, 166–167
  - starting your own, 169–187
  - surveying projects for contributions, 164–165
  - tracking projects, 167–168
  - transferring ownership, 186–187
  - triaging issues, 180–181
  - writing documentation, 178–179
  - writing `README.md` files, 178
- opening
  - branch Compare page, 137
  - browser to repository, 124
  - GitHub.com from Terminal, 137
  - pull requests, 135–140

- repo in Atom, 74

organizations (GitHub)

- about, 190
- creating, 190–191
- creating teams within, 195–196
- inviting members to, 191–192
- managing members of, 193–195
- settings for, 197–199
- using project boards within, 196–197
- viewing repositories for, 192–193

Organizations setting (GitHub.com), 23

origin, 102

Oscon (website), 288

OSD (Open Source Definition), 169

OSI (Open Source Initiative), 169

OSS (open source software)

- about, 17, 155, 169
- adding code of conduct, 173
- adding contributor guidelines, 173
- adding licenses, 170–172
- archiving projects, 185–186
- banning people, 175–176
- blocking users, 176–177
- creating open source repository, 169–173
- ending projects, 185–187
- enforcing code of conduct, 175–177
- Explore page, 156–161

  - finding places to contribute, 161–163
  - issue templates, 181–183
  - labeling issues, 179–180
  - making repositories public, 173–174
  - managing issues, 179–184
  - responding with kindness, 175
  - saved replies, 183–184
  - setting contributor expectations, 166–167
  - starting your own, 169–187
  - surveying projects for contributions, 164–165
  - tracking projects, 167–168
  - transferring ownership, 186–187
  - triaging issues, 180–181
  - writing documentation, 178–179
  - writing `README.md` files, 178
- ownership, transferring, 186–187

P

- packages (Atom), 37
- Pages (GitHub)
  - about, 64
  - creating websites with, 77–93
  - finding resources for, 76
- parent teams, creating, 199–200
- Performance and Memory tab, 315
- permissions, verifying for repo, 79–80
- personal access tokens, 25

personal info, for profile, 277–278

personalizing

- about, 251
- GitHub Actions, 260–262
- GitHub apps, 255–259
- GitHub.com accounts, 18–25
- Probot, 255–259
- using browser extensions, 251–254

picture, for profile, 277

pinned repositories

- about, 161
- on profile, 278–279

Pomodoro Technique, 312–313

Popular Topics section (Explore page), 158–160

PowerShell, 9

power-up

- defined, 220
- installing GitHub, 220–222
- using GitHub, 222–225

preferences (Atom), 37

preparing contributions, 83–90

privacy, of code, 189

Pro badge, 278

proactivity, 322–323

Probot, 255

Probot Glitch apps, creating, 256–257

profile

  - about, 32, 275–276
  - bio, 277–278
  - contribution activity, 281
  - contribution graph, 279–281
  - GitHub.com, 18–19
  - personal info, 277–278
  - picture for, 277
  - pinned repositories, 278–279
  - status message, 277

Profile setting, 197

project boards

  - about, 56–62, 316
  - creating, 56–60
  - reviewing, 81
  - using within GitHub organizations, 196–197

project management category, for tools and apps, 266

projects

  - accessing existing, 77–83
  - archiving, 185–186
  - ending, 185–187
  - milestones for, 207–208
  - orienting yourself with existing, 80–83
  - specifying, 139
  - surveying for contribution, 164–165
  - tracking, 167–168

Projects setting, 198

Projects tab, 45–46

project-specific documentation, 302–303

prototype user interfaces, 313

public setting, 17

publication, of branches, 86–89

Publish dialog box, 240

publishing category, for tools and apps, 266

publishing repositories in GitHub Desktop, 124–125

Pull Request button, 233

pull request list (GitHub Desktop), 72

pull requests

    about, 133

    adding reviewers, 138

    checking out, 230–233

    clear purpose of, 141

    commenting on code, 146–148

    creating, 139–140, 230–233, 244–246

    creating in Visual Studio, 244–246

    describing, 138

    description for, 142

    drafting, 309–310

    focus of, 141

    Git Alias and, 137

    in GitHub.com, 30

    images in, 142–143

    including call to action, 143–144

    interacting with in VS code, 237–238

    knowing your audience, 140

  - merging, 53–56
  - opening, 135–140
  - opening branch Compare page, 137
  - opening GitHub.com from Terminal, 137
  - pushing code to GitHub, 134–135
  - researching, 151
  - reviewing, 81, 144–151
  - reviewing changed files, 146
  - reviewing Conversation tab, 145
  - reviewing in Visual Studio, 244–246
  - specifying assignees, 139
  - specifying labels, 139
  - specifying projects and milestones, 139
  - submitting, 320–321
  - suggesting changes to, 148–150
  - viewing, 230–233, 244–246
  - viewing in Visual Studio, 244–246
  - writing, 140–144
- Pull requests tab, 45
- pulse, of repositories, 204
- pushing code to GitHub, 134–135
- `pwd` command, 10

Q

- `question` label, 180
- question-mark, 43
- Quick pick icon, on GitHub.com, 30

R

reading
    code of conduct, 165
    contributing code guide, 164–165
    `CONTRIBUTING.md` file, 164
    through issues, 81
`README.md` file
    modifying, 48–53
    reading, 80
    writing, 178
`ready for review` label, 180
recently added category, for tools and apps, 266
refining GitHub, 252–253
regional conferences, 286–287
Remember icon, 3
remote, 102
repositories
    about, 16
    cloning, 71–72, 99–100
    cloning in GitHub Desktop, 71–72
    Code tab, 46–47
    components of, 44–47
    creating, 113–114
    creating issues for websites, 69–71
    Explore page, 157
    exploring, 44–47

finding resources for GitHub Pages, 76

forking, 97–112, 100–112

GitHub, 35

insights of, 204–207

issues, 56–62

linking, 70, 93

making public, 173–174

merging pull requests, 53–56

modifying README.md, 48–53

open source, 38, 169–173

opening browser to, 124

opening in Atom, 74

pinned, 161, 278–279

project boards, 56–62

publishing in GitHub Desktop, 124–125

setting up, 41–44

setting up a personal website, 66–69

setting up local environment, 71–75

setup, 63–76

starring, 161, 281–282

subscribing to in Slack channel, 214–216

tabs, 45–46

top information, 44–45

tracking in GitHub Desktop, 123–124

trending, 157–158

turning a project repo into a website, 64–66

turning into a website, 64–66

  - verifying permissions for, 79–80
  - viewing for GitHub organizations, 192–193
  - website, 64, 234
- Repositories setting (GitHub.com), 23
- repository list (GitHub Desktop), 72
- Repository topics setting, 198
- researching pull requests, 151
- resolving merge conflicts, 89–90
- resources, 25, 76
- responding
  - to comments, 83
  - with kindness, 175
- responsiveness, 320
- reviewers, adding, 138
- reviewing
  - changed files, 146
  - project boards, 81
  - pull requests, 81, 144–151, 244–246
  - pull requests in Visual Studio, 244–246
- Ruby, 304–305
- running tests automatically, 311–312

S

- Saved replies setting (GitHub.com), 24
- saved reply, 183–184
- scaffolding apps, 313–314

SCM (source control management), 8

SD Ruby (website), 286

Search bar, 28

sections

- adding to personal websites, 91–92
- adding to websites, 69, 91–92

security

- for apps, 268
- reviewing code for, 146

security category, for tools and apps, 266

Security setting, 23, 198

selfies, 254

Sessions setting (GitHub.com), 23

setting(s)

- contributor expectations, 166–167
- GitHub organizations, 197–199

Settings tab, 46

setup

- collaborative coding environment, 27–38
- GitHub Desktop, 34–35
- GitHub repo, 63–76
- local environment, 71–75
- personal website repo, 66–69
- repository, 41–44

Sign in dialog box, 34–35

signing up, for GitHub.com, 17–18

Slack
    installing GitHub app for, 212–214
    subscribing to repositories in, 214–216
    trying integration with Slack, 217–219
    websites, 212
software licenses, 43
source control management (SCM), 8
Sources tab, 314
speaking, at events, 292–293
specifying
    assignees, 139
    labels, 139
    milestones, 139
    projects, 139
Squirrel icon (Atom), 36, 37
SSH key, 22–23
StackOverflow, 315
staging changes, 117–118
stale issues, closing, 83
Star button, 45
starred repositories, 161, 281–282
starting your own open source software (OSS), 169–187
`status` command, 11
status message, for profile, 277
STRIDE, 146
submitting pull requests, 320–321
subscribing to repositories in Slack channel, 214–216

suggesting changes to pull requests, 148–150

summary, in commit messages, 121

support category, for tools and apps, 266

surveying projects for contribution, 164–165

sync project button

- Atom, 75
- GitHub Desktop, 73

T

tabs, repository, 45–46

taglines

- changing for websites, 69
- modifying on personal websites, 91

team discussions, 200–201

Team for Open Source tier, 190

Team tier, 190

teams

- about, 199
- creating parent/child, 199–200
- creating within GitHub organizations, 195–196
- defined, 190

Teams setting, 198

Teams tier, 192

templates, for issues, 181–183

Terminal app

- defined, 9

  - finding, 9
  - opening GitHub.com from, 137
  - trying Git on the, 9–14
- testing
  - apps, 273–274
  - GitHub Actions, 262
- testing category, for tools and apps, 266
- TestQuality app, 274
- Things you're watching link (GitHub.com), 21
- Third-party access setting, 198
- Tip icon, 3
- titles
  - changing for websites, 69
  - changing on personal websites, 91
- track, 289
- tracking
  - projects, 167–168
  - repositories in GitHub Desktop, 123–124
- traffic, of repositories, 205
- training, in-person, 301–302
- transferring ownership, 186–187
- Trello, 219–225, 316
- trending repositories, 157–158
- triaging, 81, 82–83, 180–181
- trial and error, 297–298
- Try.NET (website), 305
- Tsc-watch for TypeScript, 312

tutorials, online, 305–306

tutorials, online coding, 304–305

Twitter, 306–307

two-factor authentication, 23, 190

U

`-u` flag, 135

Unity

- using GitHub for, 239–242
- using GitHub for Unity in, 240–242

Up for Grabs (website), 162

updates, for Atom, 36

upstream

- about, 102
- contributing changes to, 104–107
- fetching changes from, 103

user experience, for apps, 267

user groups, 286

user interface, prototype, 313

username, changing, 19

users, following, 282–283

utilities category, for tools and apps, 266

V

Verified domains setting, 198

verifying permissions for repo, 79–80

version control systems

  about, 8

  Git as, 8–16

vetting process, for GitHub Marketplace, 267–268

viewing

  issues, 233–234

  pull requests, 230–233, 244–246

  pull requests in Visual Studio, 244–246

  repositories for GitHub organizations, 192–193

Visual Git Cheat Sheet (website), 14

Visual Studio, using GitHub for, 243–246

Visual Studio Code

  about, 115, 164

  interacting with pull requests in, 237–238

  using GitHub for, 235–238

VS Code. *see* Visual Studio Code

W

waffle.io, 316

Wallaby.js for JavaScript, 312

Warning icon, 3

Watch button, 44

web hooks, 255

Webhooks setting, 198

websites

  adding sections to, 69, 91–92

adding sections to personal, 91–92

Anita Borg institute, 294

appear.in, 323

Apple source control integrations, 247

Atom, 36, 230

Atom documentation, 38

Atom Flight Manual, 38, 74, 234

Atom GitHub repo, 84

Atom packages, 37

Atom project forum, 304

Beams, Chris (blogger), 132

blocking users, 177

branching, 15

Brooklyn JS, 286

browser extensions, 252

Build, 288

“Build Your Own Octocat” app, 29

“Building an Inclusive Code Review Culture,” 151

building personal, 91–93

changing automation settings, 58

changing taglines for, 69

changing titles for, 69

Cheat Sheet, 3

VS Code, 303

Code of Conduct, 307

“Code Review Like You Mean It,” 151

CODEOWNERS, 204

Contact page, 299
creating boards and cards, 220
creating issues for, 69–71
creating with GitHub Pages, 77–93
dependency security vulnerabilities, 205
descriptions, 160
Discover page, 157
documentation example, 179
example commit message, 121
example commits, 122
Explore page, 156
Git Alias, 111
Git Docs, 14
Git documentation, 132
Git for Windows, 9
Git Merge, 292
GitHub, 4
GitHub API, 25
GitHub blog, 4, 307
GitHub Community Forum, 25, 307
GitHub Constellation, 292
GitHub contact page, 25
GitHub Contact page, 318
GitHub Desktop, 35
GitHub Desktop open source project, 80
GitHub Desktop User Guides, 72
GitHub Developers Program, 197

GitHub Education community forum, 308
GitHub for Unity extension, 242
GitHub for Visual Studio extension, 246
GitHub for VS Code pull requests extension, 238
GitHub Git Cheat Sheet, 14
GitHub Guides, 299
GitHub Help, 23, 25, 93, 298
GitHub Learning Labs, 3, 25, 300
GitHub Marketplace landing page, 268
GitHub repo, 64, 234
GitHub repository, 35
GitHub Satellite, 291
GitHub search, 282, 283
GitHub Selfie, 254
GitHub Universe, 291
git-school, 14
Glitch, 256
GoRuCo, 287
Hopper Program, 294
“How to write the perfect pull request,” 143
Jekyll, 64
JSConfHawaii, 287
Khan Academy Kids app, 306
learngitbranching, 14
“The Life of a GitHUb Action,” 260
list of apps, 258
logos, 160

Markdown GitHub Guide (website), 64

Marketplace apps, 267

Meetups.com, 286

Microsoft Imagine Cup, 287

Microsoft Open Source Code of Conduct, 165

Microsoft organization home page, 192–193

modifying taglines on personal, 91

modifying titles on personal, 91

.NET Fringe, 287

.NET São Paulo, 286

Node.js repository, 160

Octobox, 225

Octocat history, 29

Octodex of Octocats, 29

Open Source Definition (OSD), 169

open source repository, 38

Oscon, 288

outlearning labs, 76

Popular Topics section (Explore page), 158

public Slack, 304

`README.md` template, 48

release notes, 322

reporting abuse on GitHub, 318

reviewing pull requests, 81

Ruby tutorial, 304–305

for saved replies, 183

SD Ruby, 286

  - search help, 30
  - setting up PowerShell, 9
  - setting up SSD, 23
  - Slack, 212
  - software licenses, 43
  - starred repositories, 161
  - team discussion home page, 200
  - training, 302
  - Trello, 220
  - trending repositories, 157
  - Try.NET, 305
  - turning a project repo into a, 64–66
  - Up for Grabs, 162
  - version control, 8
  - Visual Git Cheat Sheet, 14
  - Visual Studio Code, 235
  - Visual Studio Code contributing guidelines, 164
  - Visual Studio Code repository, 164
  - watching repositories, 45
  - WWDC, 288
  - XKCD commit, 117
- Wiki tab, 46
- Windows, using command line in, 10
- `won't fix` label, 180
- workflow development, 309–316
- workflow integrations
  - about, 229

  - checking out pull requests, 230–233
  - creating pull requests, 230–233
  - following GitHub package for Atom, 235
  - using GitHub for Atom, 229–235
  - using GitHub for IntelliJ, 248–250
  - using GitHub for Unity, 239–242
  - using GitHub for Visual Studio, 243–246
  - using GitHub for Visual Studio Code, 235–238
  - using GitHub for XCode, 246–248
  - viewing issues, 233–234
  - viewing pull requests, 230–233
- workflows, GitHub Action, 260–262
- writing
  - blog posts, 70
  - bug reports, 318–319
  - code, 113–132
  - commit messages, 120–122
  - documentation, 178–179
  - pull requests, 140–144
  - `README.md` file, 178
- WWDC (website), 288

X

- XCode, using GitHub for, 246–248
- XKCD commit, 117

Y

Yeoman, 313–314

`yo` command, 314

Z

ZenHub, 316

About the Authors

Sarah Guthals received her PhD from UCSD in computer science specializing in CS Education in 2014. During graduate school, she built the beta version of CodeSpells, a 3D immersive video game designed to teach children to code through playing a wizard and writing spells. Sarah went on to cofound ThoughtSTEM, a company that builds software (for example, LearnToMod), curriculum, and pedagogies for teaching children to code and empowering K–12 teachers to teach their students.

Sarah has written four books around Minecraft, one on mobile app development, one on making digital games, and one on helping adults teach kids coding; launched a Coursera and EdX course for teachers interested in teaching coding; was recently named Forbes 30 under 30 in Science and UCSD's 40 under 40 Alumni; and founded We Can — a company dedicated to encouraging all kids to do anything.

Sarah has been a lecturer at UCSD for the Department of Computer Science and the Department of Educational Studies and was an engineering manager at GitHub. She is currently a senior program manager at Microsoft, focused on education.

Sarah's passion is making coding accessible to everyone, with the goal of making it a basic literacy.

Phil Haack is the founder and CEO of Haacked LLC where he coaches software organizations and helps them become the best versions of themselves.

To do this, Phil draws upon his seven years of experience at GitHub where he was a director of engineering and helped make GitHub friendly to developers on the Microsoft platform. He also draws upon his experience at Microsoft where he was a senior program manager responsible for shipping ASP.NET MVC and NuGet, among other projects. These products had permissive open source licenses and ushered in Microsoft's Open Source era.

Phil is a coauthor of the popular Professional ASP.NET MVC series and regularly speaks at conferences around the world. He's also made several appearances on technology podcasts, such as .NET Rocks, Hanselminutes, Herding Code, and The Official jQuery Podcast.

Dedications

Sarah: I would like to dedicate this book to my close friends and family members who have supported me, not only in writing this book, but in becoming who I am today. I'd like to specifically dedicate this book to my husband Adrian who has always helped me see that, with passion and dedication, I can really do anything I want. And finally to my daughter Ayla who gives me the drive to keep pushing forward!

Phil: I dedicate this book to my wife Akumi and my kids Cody and Mia who supported me immensely as I wrote this book at the same time that I was recovering from knee surgery. They fetched me many a snack and drink so I could keep writing. I also dedicate this book to the folks at GitHub past and present who changed the world of software development. Remember, the *H* is capitalized.

Authors' Acknowledgments

Sarah: I would like to acknowledge all of the hard work that went into making GitHub, all of the open source developers who share their passions on GitHub, all of the folks who build apps that integrate with GitHub, and every programming language, coding application, and effort to improve the collaborative nature of computer science — without these, coding would not be as inspiring and fun. I would also like to acknowledge Phil for c-authoring this book with me (his experience and wisdom is definitely a key feature) and Meaghan for her technical editing, making sure we don't stray. Finally, I want to thank Steve, Kelly, and the folks at Wiley for making writing this book a great experience!

Phil: GitHub changed the world of software development for the better. It's had an immense impact on how people write software today. I'm grateful to have had the opportunity to be a part of it. I want to acknowledge the founders of GitHub, the employees past and present, and the new stewards, Microsoft. I also want to acknowledge all the open source developers who were instrumental to GitHub's success. I also want to acknowledge my coauthor, Sarah, who made this an enjoyable writing experience for me. Also, thanks to Meaghan for her sharp technical editing. Finally, thank you Steve, Kelly, and the fine folks at Wiley for making this book a reality!

Publisher's Acknowledgments

Executive Editor: Steve Hayes

Project Editor: Kelly Ewing

Copy Editor: Kelly Ewing

Technical Editor: Meaghan Lewis

Editorial Assistant: Matthew Lowe

Sr. Editorial Assistant: Cherie Case

Production Editor: G. Vasanth Koilraj

Cover Image: © skynesher/Getty Images; Screenshot: Courtesy of Sarah Guthals

Take dummies with you everywhere you go!

Whether you are excited about e-books, want more from the web, must have your mobile apps, or are swept up in social media, dummies makes everything easier.

facebook
I AM A DUMMIE
For Dummies
25 YEARS OF LEARNING MADE EASY
25
25
12:34

Find us online!

dummies.com

dummies
A Wiley Brand

Leverage the power

Dummies is the global leader in the reference category and one of the most trusted and highly regarded brands in the world. No longer just focused on books, customers now have access to the dummies content they need in the format they want. Together we'll craft a solution that engages your customers, stands out from the competition, and helps you meet your goals.

Advertising & Sponsorships

Connect with an engaged audience on a powerful multimedia site, and position your message alongside expert how-to content. Dummies.com is a one-stop shop for free, online information and know-how curated by a team of experts.

- Targeted ads
- Video
- Email Marketing
- Microsites
- Sweepstakes sponsorship

25 YEARS OF LEARNING MADE EASY
25
25

20 MILLION PAGE VIEWS EVERY SINGLE MONTH

15 MILLION UNIQUE VISITORS PER MONTH

43% OF ALL VISITORS ACCESS THE SITE VIA THEIR MOBILE DEVICES

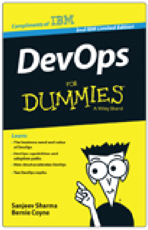
700,000 NEWSLETTER SUBSCRIPTIONS TO THE INBOXES OF 300,000 UNIQUE INDIVIDUALS EVERY WEEK

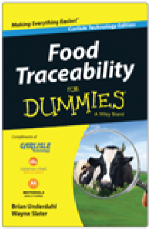
of dummies

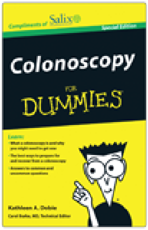
Custom Publishing

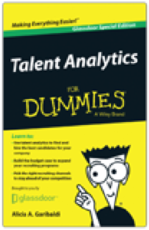
Reach a global audience in any language by creating a solution that will differentiate you from competitors, amplify your message, and encourage customers to make a buying decision.

- Apps
- Books
- eBooks
- Video
- Audio
- Webinars


DevOps
FOR
DUMMIES
Sanjeev Sharma
Bernie Coyne


Food
Traceability
FOR
DUMMIES
Brian Unterdahl
Wayne Slater


Colonoscopy
FOR
DUMMIES
Kathleen A. Doble


Talent Analytics
FOR
DUMMIES
Alicia A. Garibaldi

Brand Licensing & Content

Leverage the strength of the world's most popular reference brand to reach new audiences and channels of distribution.

For more information, visit dummies.com/biz

dummies
A Wiley Brand

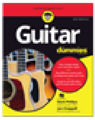
PERSONAL ENRICHMENT

Staying Sharp
dummies

9781119187790
USA $26.00
CAN $31.99
UK £19.99

Facebook
dummies

9781119179030
USA $21.99
CAN $25.99
UK £16.99


Guitar
dummies

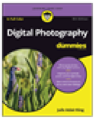
9781119293354
USA $24.99
CAN $29.99
UK £17.99

Investing
dummies

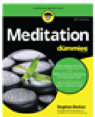
9781119293347
USA $22.99
CAN $27.99
UK £16.99

Beekeeping
dummies

9781119310068
USA $22.99
CAN $27.99
UK £16.99


Digital Photography
dummies

9781119235606
USA $24.99
CAN $29.99
UK £17.99


Meditation
dummies

9781119251163
USA $24.99
CAN $29.99
UK £17.99

Pregnancy
ALL-IN-ONE
dummies
6 Books

9781119235491
USA $26.99
CAN $31.99
UK £19.99

Samsung Galaxy S7
dummies

9781119279952
USA $24.99
CAN $29.99
UK £17.99

iPhone
dummies

9781119283133
USA $24.99
CAN $29.99
UK £17.99

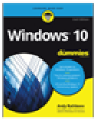
Crocheting
dummies

9781119287117
USA $24.99
CAN $29.99
UK £16.99

Nutrition
dummies

9781119130246
USA $22.99
CAN $27.99
UK £16.99

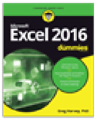
PROFESSIONAL DEVELOPMENT


Windows 10
dummies

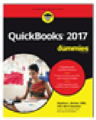
9781119311041
USA $24.99
CAN $29.99
UK £17.99

AutoCAD
dummies

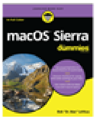
9781119255796
USA $39.99
CAN $47.99
UK £27.99


Microsoft
Excel 2016
dummies

9781119293439
USA $26.99
CAN $31.99
UK £19.99


QuickBooks 2017
dummies

9781119281467
USA $26.99
CAN $31.99
UK £19.99


macOS Sierra
dummies

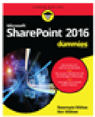
9781119280651
USA $29.99
CAN $35.99
UK £21.99

LinkedIn
dummies

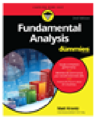
9781119251132
USA $24.99
CAN $29.99
UK £17.99

Windows 10
ALL-IN-ONE
dummies
10 Books

9781119310563
USA $34.00
CAN $41.99
UK £24.99


Microsoft
SharePoint 2016
dummies

9781119181705
USA $29.99
CAN $35.99
UK £21.99


Fundamental Analysis
dummies

9781119263593
USA $26.99
CAN $31.99
UK £19.99

Networking
dummies

9781119257769
USA $29.99
CAN $35.99
UK £21.99

Microsoft
Office 2016
dummies

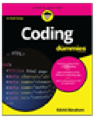
9781119293477
USA $26.99
CAN $31.99
UK £19.99

Microsoft
Office 365
dummies

9781119265313
USA $24.99
CAN $29.99
UK £17.99

Salesforce.com
dummies

9781119239314
USA $29.99
CAN $35.99
UK £21.99


Coding
dummies

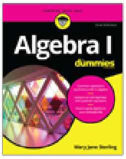
9781119293323
USA $29.99
CAN $35.99
UK £21.99

dummies.com

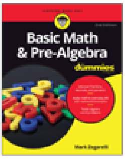
dummies®
A Wiley Brand

Learning Made Easy

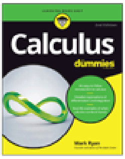
ACADEMIC


Algebra I
dummies

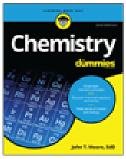
9781119293576
USA $19.99
CAN $23.99
UK £15.99


Basic Math & Pre-Algebra
dummies

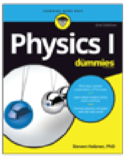
9781119293637
USA $19.99
CAN $23.99
UK £15.99


Calculus
dummies

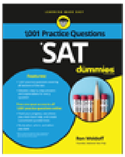
9781119293491
USA $19.99
CAN $23.99
UK £15.99


Chemistry
dummies

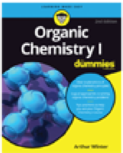
9781119293460
USA $19.99
CAN $23.99
UK £15.99


Physics I
dummies

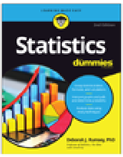
9781119293590
USA $19.99
CAN $23.99
UK £15.99


1,001 Practice Questions
SAT
dummies

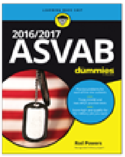
9781119215844
USA $26.99
CAN $31.99
UK £19.99


Organic Chemistry I
dummies

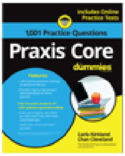
9781119293378
USA $22.99
CAN $27.99
UK £16.99


Statistics
dummies

9781119293521
USA $19.99
CAN $23.99
UK £15.99


2016/2017
ASVAB
dummies

9781119239178
USA $18.99
CAN $22.99
UK £14.99


Includes Online Practice Tests
1,001 Practice Questions
Praxis Core
dummies

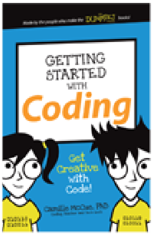
9781119263883
USA $26.99
CAN $31.99
UK £19.99

Available Everywhere Books Are Sold

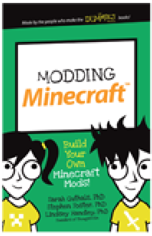
dummies.com

dummies
A Wiley Brand

Small books for big imaginations


GETTING STARTED WITH Coding
Get Creative with Code!

9781119177173
USA $9.99
CAN $9.99
UK £8.99


MODDING Minecraft
Build Your Own Minecraft Mods!

9781119177272
USA $9.99
CAN $9.99
UK £8.99

MAKING YouTube VIDEOS
Star in Your Own Video!

9781119177241
USA $9.99
CAN $9.99
UK £8.99

DESIGNING Digital Games
Create Games with Scratch!

9781119177210
USA $9.99
CAN $9.99
UK £8.99

GETTING STARTED WITH Raspberry Pi
Program Your Raspberry Pi!

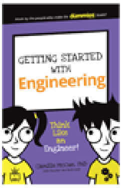
9781119262657
USA $9.99
CAN $9.99
UK £6.99

EXPERIMENTING WITH Science
Think, Test, and Learn!

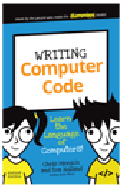
9781119291336
USA $9.99
CAN $9.99
UK £6.99

CREATING Digital Animations
Animate Stories with Scratch!

9781119233527
USA $9.99
CAN $9.99
UK £6.99


GETTING STARTED WITH Engineering
Think Like an Engineer!

9781119291220
USA $9.99
CAN $9.99
UK £6.99


WRITING Computer Code
Learn the Language of Computers!

9781119177302
USA $9.99
CAN $9.99
UK £8.99

Unleash Their Creativity

dummies.com

dummies
A Wiley Brand

Take Dummies with you everywhere you go!

Go to our Website

Like us on Facebook

Follow us on Twitter

Watch us on YouTube

Join us on LinkedIn

Pin us on Pinterest

Subscribe to our newsletter

Create your own Dummies book cover

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley's ebook EULA.